UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matevž Kovačič

AVTOMATIZIRANO KADROVANJE ZA OBLIKOVANJE VIRTUALNEGA TIMA

MAGISTRSKO DELO

Mentor: doc. dr. Marko Bajec

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št.: 127-MAG-RI/2009
Datum: 23. 4. 2009

Matevž KOVAČIČ, univ. dipl. mat.

Ljubljana

Fakulteta za računalništvo in informatiko Univerze v Ljubljani izdaja naslednjo magistrsko nalogo

Naslov naloge: **Avtomatizirano kadrovanje za oblikovanje virtualnega tima**

Automatic personnel selection for virtual team design

Tematika naloge:

Kandidat naj v magistrski nalogi predstavi pojem virtualnih timov, potrebo po virtualnih timih in njihovo oblikovanje. Naredi naj analizo funkcionalnosti portala borze dela, sestavi podatkovni model in sprojektira ogrodje spletnega portala v trinivojski arhitekturi. Pri tem naj uporablja sodobno Microsoftovo tehnologijo: ASP.NET 2.0, podatkovno bazo SQL Server 2005 in ostale dodatke, ki pripomorejo k učinkovitejšemu in standardiziranemu programiranju in oblikovanju.
Kandidat naj identificira atribute s katerimi se okarakterizira iskalce dela. Zgradi naj model večparametrskega odločanja na podlagi katerega se bo izvedlo ocenjevanje iskalcev dela. Razišče naj več metod večparametrskega odločanja, ki bi bile primerne za reševanje problema. Glede na oceno iskalcev dela naj se izbere optimalen tim, pri tem pa naj uporabi optimizacijsko metodo celoštevilskega programiranja.
Glavni prispevek magistrskega dela naj bo razvoj celovitega sistema za avtomatizirano sestavo virtualnega tima, ki najbolj ustreza potrebam projekta.

Mentor:

doc. dr. Marko Bajec

UNIVERZA V LJUBLJANI • Fakulteta za računalništvo in informatiko •

Dekan:

prof. dr. Franc Solina

I Z J A V A O A V T O R S T V U

magistrskega dela

Spodaj podpisani/-a Matevž Kovačič,

z vpisno številko 63020346,

sem avtor/-ica magistrskega dela z naslovom

Avtomatizirano kadrovanje za oblikovanje virtualnega tima

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal/-a samostojno, pri delu me je vodil mentor (naziv, ime in priimek) doc. dr. Marko Bajec;
- so elektronska oblika magistrskega dela, naslova (slov., angl.), povzetka (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko magistrskega dela;
- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki »Dela FRI«.

V Ljubljani, _____ Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se mentorju doc. dr. Marku Bajcu za nasvete in pomoč pri izdelavi magistrske naloge ter članom Laboratorija za informatiko, ki so pripomogli pri izvedbi sistema za vodenje borze dela.

Kazalo

1 UVOD 11

1.1 OPIS PROBLEMA IN CILJI 11
1.2 VSEBINA MAGISTRSKE NALOGE 12
1.3 PREGLED LITERATURE IN RAZISKAV 13

2 VIRTUALNI TIMI 15

2.1 KAJ SO VIRTUALNI TIMI 15
2.2 PRIPRAVA VIRTUALNEGA TIMA 16
2.3 ATRIBUTI KANDIDATOV ZA TIM 16
2.3.1 Splošni podatki 17
2.3.2 Strokovna usposobljenost 18
2.3.3 Ocena delovne uspešnosti 19
2.3.4 Tehnični kriteriji 23

3 SISTEM ZA VODENJE BORZE DELA 25

3.1 ARHITEKTURA PORTALA 25
3.2 UPORABLJENA TEHNOLOGIJA 26
3.2.1 ASP.NET Spletne aplikacije 26
3.2.2 SQL Strežnik 28
3.2.3 NHibernate 29
3.3 FUNKCIONALNOST PORTALA 30
3.4 ENTITETNI MODEL 41

4 LINEARNO PROGRAMIRANJE IN VEČPARAMETRSKI ODLOČITVENI PROBLEMI 47

4.1 METODOLOGIJA OPERACIJSKIH RAZISKAV 47
4.2 FORMULACIJA LINEARNEGA PROGRAMA 49
4.3 REŠEVANJE LP 51
4.4 ODLOČANJE 52
4.5 MERJENJE KORISTNOSTI IN LOGIKA ODLOČANJA 54
4.6 VEČPARAMETRSKO ODLOČANJE 55
4.7 OPISLJIVOST VARIANT 58
4.8 STRUKTURIRANJE PARAMETROV 59
4.9 FAZE ODLOČITVENEGA POSTOPKA 60
4.10 AHP METODA 63
4.11 MAUT METODA 65

5 MODEL ZA SESTAVO VIRTUALNEGA TIMA 67

5.1 IDENTIFIKACIJA ODLOČITVENEGA PROBLEMA 67
5.2 IDENTIFIKACIJA KRITERIJEV 67
5.3 ZAJEMANJE ODLOČITVENEGA ZNANJA 70
5.4 VREDNOTENJE IN ANALIZA VARIANT 70

5.5 PROBLEM LINEARNEGA PROGRAMIRANJA 72
5.6 IMPLEMENTACIJA ISKALNEGA ALGORITMA 73
6 PORTAL ZA KADROVANJE 77
7 ZAKLJUČEK 90
PRILOGA A 91
VIRI IN LITERATURA 99

Kazalo slik

SLIKA 3-1: ARHITEKTURA REŠITVE 25
SLIKA 3-2: ARHITEKTURA SQL STREŽNIKA 28
SLIKA 3-3: PRIMER UPORABE ZA KADROVSKI PORTAL 32
SLIKA 3-4: ZASLONSKA MASKA 38
SLIKA 3-5: ENTITETNI MODEL ZA KADROVSKI PORTAL 42
SLIKA 4-1: PROCES VEČPARAMETRSKEGA ODLOČANJA 56
SLIKA 4-2: VEČPARAMETRSKI ODLOČITVENI MODEL 57
SLIKA 4-3: HIERARHIČNA STRUKTURA KRITERIJEV 63
SLIKA 5-1: HIERARHIČNA STRUKTURA ZA GLAVNI PARAMETER IN PRVE LOKALNE KRITERIJE 68
SLIKA 5-2: HIERARHIČNA STRUKTURA ZA PODKRITERIJ SPLOŠNI PODATKI 68
SLIKA 5-3: HIERARHIČNA STRUKTURA ZA PODKRITERIJ STROKOVNA USPOSOBLJENOST 69
SLIKA 5-4: HIERARHIČNA STRUKTURA ZA PODKRITERIJ DELO IN OSEBNE LASTNOSTI 69
SLIKA 5-5: HIERARHIČNA STRUKTURA ZA PODKRITERIJ TEHNIČNI KRITERIJI 70
SLIKA 5-6: OCENJEVANJE ALTERNATIV PO HIERARHIJI 71
SLIKA 6-1: VNOSNA MASKA ZA ŽIVLJENJEPIS 77
SLIKA 6-2: VNOS KRITERIJEV ZA PROJEKT 78

Kratice, okrajšave, simboli

AHP	Analitični hierarhični proces, tudi metoda AHP (*Analytic Hierarchial Process*)
MAUT	Teorija večparametrske koristnosti (*Multi-Attribute Utility Theory*)
ASP.NET	Ogrodje za spletne aplikacije, s katerim je mogoče graditi dinamične spletne strani (*Active Server Pages*) s pomočjo tehnologije .NET
ORM	Programerska tehnika za preslikavo med relacijsko bazo in objekti (*Object Relational Mapping*)
XML	Razširljiv označevalni jezik (*eXtensible Markup Language*)
.NET	*Microsoft .NET* je ogrodje za razvijanje programske opreme na Microsoft Windows operacijskih sistemih
SMTP	Preprost protokol za prenos elektronske pošte (*Simple Mail Transfer Protocol*)
CV	Kratek, jedrnat življenjepis (*Curriculum Vitae*)
CSS	Predloge, ki določajo izgled spletnih strani (*Cascading Style Sheets*)
LP	Linearno programiranje (*Linear Programming*)

Povzetek

Magistrska naloga predstavi pojem virtualnih timov, potrebo po virtualnih timih in njihovo oblikovanje. Narejena je analiza funkcionalnosti portala borze dela, sestavljen podatkovni model in sprojektirano ogrodje spletnega portala v trinivojski arhitekturi. Pri tem je uporabljena sodobna Microsoftova tehnologija: ASP.NET 2.0, podatkovna baza SQL Server 2005 in ostali dodatki, ki pripomorejo k učinkovitejšemu in standardiziranemu programiranju in oblikovanju.

V nalogi so identificirani atributi, s katerimi se okarakterizira iskalce dela. Zgrajen je model večparametrskega odločanja, na podlagi katerega se izvede ocenjevanje iskalcev dela. Raziskanih je več metod večparametrskega odločanja, ki bi bile primerne za reševanje problema. Glede na oceno iskalcev dela se izbere optimalen tim, pri tem pa je uporabljena optimizacijska metoda celoštevilskega programiranja.

Glavni prispevek magistrskega dela je razvoj celovitega sistema za avtomatizirano sestavo virtualnega tima, ki najbolj ustreza potrebam projekta.

Ključne besede: virtualni tim, management človeških virov, izbira kadrov, kadrovski informacijski sistem, večparametrsko odločanje, optimizacijske metode, celoštevilsko programiranje.

Abstract

The following Master's thesis examines the concept of virtual teams, the need for virtual teams and how they are designed. An analysis of the functionality for human resource information system has been made and a data model has been projected. For a web portal has been used three-layer architecture. For the portal implementation advanced Microsoft technology has been used: ASP.NET 2.0, SQL Server 2005 database and other third-party software which contribute to a more efficient and standard programming and design.

The thesis also describes variables used for multi-criteria decision making regarding job seekers. The core of the thesis is the personnel selection model, on behalf of which the assessment of job seekers is performed. Several methods for multi-criteria decision making have been described and the most appropriate method for solving the problem has been determined. According to the job seekers assessment the optimal team is selected by using the integer programming optimisation method.

The main contribution of this thesis is the development of a comprehensive system for the automatic personnel selection for designing a virtual team which best suits the needs of the project.

Keywords: virtual team, human resource management, personnel selection, human resource information system, multi-criteria decision making, optimisation methods, integer programming

1 UVOD

1.1 Opis problema in cilji

Dandanes se programerske hiše pogosto soočajo z zaposlovanjem pravilnega kadra, ki ustreza hitrim spremembam v razvojnih tehnologijah in spreminjajočim se projektnim trendom. Znanje kadra v programerskih tehnologijah je hitro zastarano in nemogoče je, da en človek znova in znova obnavlja svoje strokovno znanje na svojem področju in ostaja v stiku z najnovejšo tehnologijo. Prav tako pa je potrebno ohraniti stara znanja, saj razviti produkti zahtevajo vzdrževanje po več let. Klasična rešitev je začasno zaposlovanje novih ljudi ter izobraževanje že zaposlenih, katerih znanje je že preveč zastarano. Težava je, da je težko dobiti dovolj novega kadra z zahtevanim novim znanjem. Izobraževanje ljudi, ki so izgubili stik z moderno tehnologijo, je drago in težavno. Ob koncu projekta pa se nam sprostijo zaposleni, ki nam niso več potrebni.

Alternativa tej rešitvi je sestava virtualnega tima. Glavna prednost te rešitve je, da kader izberemo iz velikega kroga iskalcev dela, ki imajo točno zahtevana znanja in sposobnosti. Virtualni tim se sestavi za izdelavo točno določenega projekta in se po končanju razpusti.

Kandidate za sestavo virtualnega tima se lahko išče preko internetnega portala, kjer ima vsak kandidat možnost vpisa svojih karakteristik. Ponudniki pa vnesejo zahteve in potrebe projektov. Na podlagi teh potreb se avtomatično pošljejo ponudbe kandidatom, ki bi optimalno sestavljali tim.

Glavni namen naloge je torej prototipno izdelati portal borze dela, kjer bodo imeli iskalci dela možnost vnosa svojih podatkov, ponudniki dela pa možnost kreiranja virtualnih timov na podlagi potreb projekta. Kreiranje tima bo predstavljeno kot računalniško podprt večparametrski hierarhični model procesa izbire kadrov v kombinaciji z optimizacijsko metodo.

1.2 Vsebina magistrske naloge

V drugem poglavju je podrobneje opisan pojem virtualnih timov in sestavljanje virtualnih timov. Identificirani so atributi, s katerimi okarakteriziramo iskalce dela s poudarkom na področju računalništva. Za pridobitev vrednosti teh atributov za posameznega iskalca se je naredil portal borze iskalcev dela.

V naslednjem poglavju je opisana teoretična arhitektura spletnega portala in uporabljena tehnologija za implementacijo. Narejena je analiza osnovne funkcionalnosti spletnega portala za kadrovski sistem na podlagi podobnih zaposlitvenih portalov. Na podlagi analize je definiran podatkovni model kadrovskega sistema. Za podatkovno bazo je uporabljen SQL Server 2005. Arhitektura ogrodja portala je trinivojska spletna aplikacija. Uporabniški vmesnik temelji na Microsoftovi tehnologiji ASP.NET 2.0, podatkovni nivo pa je realiziran z NHibernate rešitvijo.

Četrto poglavje opredeljuje odločanje in sisteme za pomoč odločanju. Opisane so osnovne metode in metode večparametrskega modeliranja. Podrobneje sta opisani metodi večparametrskega odločanja MAUT in AHP, ki sta bili primerni za reševanje problema. Opredeljena so tudi različna področja uporabe. Predstavljena sta tudi teoretična formulacija linearnega programiranja in njegovo reševanje.

V petem poglavju je na osnovi teoretičnih spoznanj drugega in četrtega poglavja zgrajen model večparametrskega odločanja za rangiranje kandidatov. Glede na literaturo, omenjeno v prejšnjem podpoglavju, se za sestavo virtualnega tima uporabi kombinacijo metod večparametrskega odločanja in optimizacijsko metodo. Za optimizacijo uporabimo eno najpopularnejših tehnik operacijskih raziskav: linearno programiranje.

AHP metoda večparametrskega odločanja je ocenjevalna metoda, ki vizualno strukturira kompleksne odločitvene probleme. Temelji na dekompoziciji kriterijev, medsebojnih primerjanj in združevanju priorite. AHP pa je tudi teorija merjenja kriterijev, ki so številčno nemerljivi. S pomočjo AHP metode dobimo uteži, ki predstavljajo relativno pomembnost teh atributov.

MAUT metoda obsega kreiranje funkcij koristnosti za posamezne atribute in kasnejše združevanje ocen po atributih v vrednosti (uteži) na nivoju višje. Naš model atributov za virtualni tim ima trinivjosko drevesno strukturo. Najnižji tretji nivo ima za podredna vozlišča liste drevesa.

Za optimalno izbiro tima ne bomo izbirali kadra po atributih v listih, ampak po vrednostih v vozliščih na tretjem nivoju, ki predstavljajo bolj zaokrožena področja različnih znanj. Najprej naredimo na tem nivoju selekcijo. Glede na velikost tima lahko izločimo kandidate, ki so po vseh kriterijih slabši od n drugih kandidatov, kjer je n velikost tima. S tem zmanjšamo krog kandidatov za vhodne podatke v problem celoštevilskega programiranja. Utež kandidata na tretjem nivoju določa vpliv, ki ga ima enota odločitvene spremenljivke na namensko funkcijo in omejitev v tem problemu. Omejitve določimo glede na maksimalno število področij, ki jih

lahko zavzema kandidat, maksimalno število kandidatov, ki jih je lahko na eno področje in število kandidatov v timu. Rešitev tega problema nam prinese optimalen izbor kandidatov za virtualni tim.

Spletni portal za kadrovski sistem nam poda sistem za enostaven in slikovit opis potreb projekta po določenem kadru. Skupaj z razvito kombinacijo različnih metod večparametrskega odločanja in modelom celoštevilčnega programiranja pa prinese učinkovito in celovito aplikacijo za avtomatiziran izbor virtualnega tima na podlagi kandidatov, ki oddajo svoje ocene karakteristik.

1.3 Pregled literature in raziskav

Upravljanje z virtualnimi timi je predstavljeno v članku [6], kjer avtor opiše celoten ciklus od priprave tima, zagona, vplivanja na učinkovitost, razvoja skupine, do razpustitve virtualnega tima. Glavni poudarek raziskave je kvantitativna raziskava že obstoječih virtualnih timov v organizacijah in podani povzetek ciklusa virtualnega tima. Iz izsledkov so podana razna priporočila za upravljanje s človeškimi viri. Podobna raziskava na 12 virtualnih timih je izvedena v članku [13]. Podani so kriteriji, ki vplivajo na učinkovitost tima, vodenje dela v timu in zadovoljstvo tima. Teoretične osnove določanja potrebnih človeških zmožnosti, pridobivanja delavcev, izbirnih postopkov in preizkusov ter uvajanja delavcev so podani v doktorski disertaciji [8]. Posebej so opisani dejavniki delovne uspešnosti ter osnove ocenjevanja in spremljanja delovne uspešnosti. Ogrodje za metodološki pristop, ki bi podjetjem omogočilo razvoj programske opreme s pomočjo virtualnih timov, je podano v prispevku [20].

Model izbire kadra s pomočjo večparametrskega modela je predstavljen v literaturi [9], kjer je podan večnivojski model atributov za karakterizacijo kadra. V članku [22] je podan pogled na virtualni tim kot na poenostavljeno virtualno organizacijo. Podobno kot pri virtualni organizaciji je virtualni tim karakteriziran glede na člane, ki ga sestavljajo. Člani so šibko povezani in fizično razkropljeni, razlikujejo se kulturno in organizacijsko. Model virtualnega tima, ki je razvit, je namenjen preučitvi vpliva posameznih virtualnih karakteristik na učinkovitost tima. Pregled karakteristik dobaviteljev v člankih od leta 1992 do 2003 in povzetek metod za izbiro dobaviteljev je narejen v članku [23].

Veliko literature priporoča uporabo večparametrskega modela za reševanje problema izbora kadra. Izbira partnerjev po večparameterskem modelu je opisana v članku [18] s kombinacijo AHP in MAUT metode in s pomočjo celoštevilskega programiranja. Za formacijo večfunkcionalnih timov [10] se uporabi AHP metoda za generiranje podatkov za optimizacijo. Podobno kot prej se tudi tukaj uporabi model celoštevilskega programiranje za formiranje timov. AHP metoda se tudi uporabi za ovrednotenje kriterijev v večnivojskem modelu in olajša izbor najprimernejšega ponudnika programske opreme [3]. Metoda, ki se priporoča [11] za izbor najprimernejšega projekta za programersko hišo, pa priporoča uporabo kombinacij

metod Delphi, AHP in ciljnega programiranja (goal programming). Metoda AHP, ki je ena najbolj znanih in popularnih metod, je bila razvita s strani Thomasa L. Saaty-ja v sedemdesetih letih. Od takrat se je razvilo ogromno razširitev in izboljšav, metoda pa temelji na nekaj temeljnih aksiomih, podanih v njegovi knjigi [17]. Kreiranje optimalnega virtualnega tima se rešuje z linearnim programiranjem [2], zaradi posebnosti primera pa uporabimo posebno vrsto linearnega programiranja: celoštevilčno programiranje [10, 21].

2 VIRTUALNI TIMI

2.1 Kaj so virtualni timi

Razdelitev opravljanja dela na različno oddaljene lokacije in v različnih časovnih pasovih ni več nekaj posebnega že zadnjih 15 let. Obstaja veliko poučnih primerov, kako ljudje sodelujejo na velike razdalje ne glede na časovne razlike. V času hitrega razvoja prenosa informacij in komunikacije je porazdeljeno delo postalo enostavnejše, hitrejše in učinkovitejše. Pridevnik virtualni pa je označil porazdeljeno delo, ki temelji na orodjih, ki izkoriščajo prenos informacij in komunikacije.

Na splošno lahko razlikujemo virtualno delo glede na število vpletenih akterjev in stopnjo medsebojne interakcije. Prvi primer je telework, kjer se dela delno ali v celoti izven nosilnega podjetja. Komunikacija poteka s pomočjo telekomunikacijskih storitev. Virtualne skupine nastanejo kot kombinacija več telework skupin in ko vsak član take skupine poroča istemu nadrejenemu.

Nasprotno pa je drugi primer, virtualni tim, ko vsi člani skupine komunicirajo med seboj z namenom doseči isti cilj. Skupino poimenujemo *virtualni tim*, če zadovolji naslednjim zahtevam [22]:

a) skupina sestoji iz dveh ali več članov;
b) člani sodelujejo med seboj interaktivno z namenom doseči isti cilj;
c) vsaj en član skupine se nahaja na drugi lokaciji, v drugi organizaciji ali v drugem časovnem pasu;
d) komunikacija in koordinacija poteka predvsem preko elektronskih medijev.

Predvsem zadnji dve zahtevi sta bistveni zahtevi, ki ločita virtualni tim od navadnega tima ljudi. Virtualnost tima lahko merimo z razlikovanjem od navadnega tima: povprečna razdalja med dvema članoma, število delovnih mest, ki so na različnih lokacijah, glede na število vseh delovnih mest.

Kot v vseh drugih skupinah lahko tudi v virtualnih timih z visoko virtualnostjo preučujemo posledice take sestave na nivoju individualcev, na nivoju orgranizacije in na sociološkem nivoju. Prednosti na individualnem nivoju so večja prilagodljivost in držanje roka, večja odgovornost in večja motiviranost. Izzivi na drugi strani pa so večji občutek izolacije posameznika in manjši osebni stiki. Večja je možnost nerazumevanja naloge in napačno porazdeljevanje nalog in vlog posameznikom. Na nivoju organizacije imajo virtualni timi predvsem strateške prednosti. Time se lahko sestavi iz ekspertov, ki niso samo iz lokalnih logov. Tim se lahko zgradi z viri iz vseh časovnih pasov, kar lahko pomeni delo 24 ur na dan. Hitrost in prilagodljivost organizacije na zahteve tržišča se lahko povečata, zmanjšajo se tudi stroški potovanj in stroški zaposlitve ter delovnega mesta. To pa lahko povzroči težave pri

planiranju, da ne pride do neproduktivnega delovnega časa, prav tako so stroški s tehnologijo za komunikacijo in težave z varnostjo prenešenih podatkov. Na sociološkem nivoju je prednost predvsem možnost, da se v time lahko vključi kandidate s področij s slabšo infrastrukturo in večjo brezposelnostjo. S tem se tudi pripomore k regionalnemu razvoju. Ni ovir, da se ne bi vključilo ljudi s telesnimi hibami, ki se ne morejo vključiti v normalne službe. Ker ni migracije med domom in službo, se zmanjša težava transporta in posledično onesnaženja zraka.

2.2 Priprava virtualnega tima

Najpomembnejša naloga pri pripravi virtualnega tima je točna definicija namena tima, ugotoviti je potrebno nivo virtualnosti, ki bi bil najprimernejši za dosego zastavljenega cilja [6]. Te odločitve ponavadi temeljijo na različnih strateških faktorjih, od potreb na tržišču, potreb po zmanjševanju stroškov, do potreb po združevanju tehnologij. Ko so te odločitve sprejete, je potrebno določiti potrebo kadra, določiti naloge in sistem nagrajevanja ter tehnologijo, ki se bo uporabljala. Nato pa se preide v sestavo virtualnega tima.

Ena izmed glavnih strategij pri virtualnih timih je sestaviti skupino sposobnih specialistov z različnih lokacij. V tem primeru je glavni kriterij pri izbiri kadra njegovo tehnično in strokovno znanje. Dodatno pa se preverjajo še splošni atributi kadra. Te delimo v tri glavne skupine:

a) splošna razgledanost;
b) delovne navade (vestnost, pravilnost);
c) skupinsko delo (osebna stabilnost, odprtost, samostojnost).

Večja kot je virtualnost skupine, večja je potreba po tehničnem znanju komuniciranja, po samoorganiziranosti, iznajdljivosti in neodvisnosti. Prav tako so lahko člani skupin kulturno zelo različni in prav ta raznolikost je marsikdaj povezana s sinergijskim učinkom. Raziskave razlik med navadnimi in raznolikimi timi poudarjajo, da je raznolikost predvsem prednost pri težkih in kompleksnih problemih, ki niso dobro definirani in zahtevajo kreativnost.

2.3 Atributi kandidatov za tim

Do dobrih ali slabih odločitev pri izbiri najpogosteje prihaja zaradi razlike med usposobljenostjo in ustreznostjo kandidata. Čim manjši je odklon med usposobljenostjo in ustreznostjo, tem boljša je odločitev. Ko v procesu izbire kandidata za določeno delo ugotavljamo njegovo usposobljenost, moramo ugotoviti, ali ima kandidat ustrezne kvalifikacije in delovne izkušnje za izvajanje tega dela. Ko pa ugotavljamo njegovo ustreznost, moramo ugotoviti, ali ima ustrezne lastnosti za izvajanje tega dela, ali se bo ujemal s sodelavci, podobo dela in vodjo. Verjetnost prave izbire je večja, če damo prednost kandidatom, ki najbolj ustrezajo. Usposobljenost sestavljajo znanja, spretnosti in izkušnje in jih lahko povečujemo ali dopolnjujemo z izobraževanjem in usposabljanjem. Veliko težje, če že ne nemogoče, je vplivati na osebnostne lastnosti in kvalitete, torej na ustreznost posameznika za določeno delo.

Lastnosti delavcev lahko razčlenimo na motivacijo delavcev. Določi se, kakšen mora biti kandidat, da bo zares želel opravljati določeno delo, to je, kaj bi moralo biti kandidatu pri delu

všeč in česa naj ne bi maral ter kaj naj bi kandidat vrednotil, katere interese imel in kaj želel doseči, da bi mu organizacija to lahko pri njegovem delu tudi zares ponudila.

Drug pomemben element so kandidatove sposobnosti, to je, kakšno znanje naj bi imel, kakšne posebne spretnosti in kakšne delovne ali študijske dosežke, ki bi nakazovali, da bo lahko dosegal delovne cilje. Kandidatove zmožnosti naj bi ustrezale strukturi dela, ki naj bi ga opravljal. Tretja razsežnost posameznika je njegov stil, ki se opredeljuje z osebnostnimi lastnostmi, fizičnimi značilnostmi in socialnimi vezmi oziroma načinom zasebnega življenja. Stil kandidata naj bi se ujemal z delovnim okoljem, v katerem delo poteka.

Z modelom za izbiro kadrov smo opredelili tudi kriterije, na podlagi katerih lahko izbiramo med več kandidati tiste, ki najbolj ustrezajo zahtevam določenega dela. Glavni sklopi kriterijev, ki pogojujejo izbor ustreznosti kandidata za določeno delovno mesto so naslednji [8,6,3]:

- splošni podatki,
- strokovna usposobljenost,
- ocena delovne uspešnosti in osebnostne lastnosti,
- tehnično znanje.

2.3.1 Splošni podatki

Spol: moški, ženski

Za opravljanje različnih del se pojavlja tudi zahteva oziroma potreba po določenem spolu, na primer: za težka fizična dela so primernejši moški in se za to zahteva moško delovno silo, medtem ko za opravljanje tajniških del organizacije v glavnem dajejo prednost ženskam.

Bivanje: Slovenija, Balkan, Evropa, Amerika, Azija, ...

Nekatere organizacije zaposlujejo samo določene državljane oziroma ne zaposlujejo delavcev iz določenih držav zaradi slabih izkušenj in različnih problemov oziroma težav, ki so s tem povezane. Ker je poudarek na virtualizaciji in se člani lahko nahajajo po celem svetu, omejitev ne sme biti premočna. Omejitve zato postavimo na domačo državo, širšo regijo, Evropo oz. druge kontinente.

Starost: 18-27, 28-37, 38-47, 48-57, 58-67, 68-

Tudi starost je povezana z vrsto dela, za katerega potrebujemo delavca. V primeru, ko gre za zaposlovanje novega delavca, organizacije raje izbirajo med mlajšimi delavci. Vendar je to v tem primeru odvisno od dela, za katerega potrebujejo delavca, oziroma od tega, ali potrebujejo nekvalificiranega ali kvalificiranega delavca, ali potrebujejo vodilnega delavca itn. Ko pa izbirajo med že zaposlenimi delavci, raje izbirajo med tistimi, ki so že dolgo časa v podjetju oziroma organizaciji, ker jo le-ti dobro poznajo in imajo bogate izkušnje.

Cena: drag, srednje drag, normalna cena, sredenje poceni, poceni

Cena kandidata je faktor, ki pogosto vpliva na izbor kandidata. Odločitev, ali vzeti boljšega in dražjega kandidata, je večkrat na preizkušnji. Marsikdaj pa tudi cena ni glavni faktor pri odločitvi.

2.3.2 Strokovna usposobljenost

Pod pojmom strokovna usposobljenost delavca razumemo vsoto vseh znanj in izkušenj, ki jih mora imeti posameznik, da bi lahko učinkovito opravljal svoje delo. Kakovost strokovne usposobljenosti opredeljujejo podatki o stopnji izobrazbe, smeri izobrazbe, poklicu, znanju tujih jezikov, funkcionalnih znanjih in delovnih izkušnjah.

***Stopnje izobrazbe*:**

1. stopnja: nepopolna osnovna šola,
2. stopnja: osnovna šola in tečaji,
3. stopnja: 2-letna srednja izobrazba,
4. stopnja: 3-letna srednja izobrazba,
5. stopnja: 4-letna srednja izobrazba,
6. stopnja: višja izobrazba,
7. stopnja: visoka izobrazba,
8. stopnja: magisterij ali specializacija,
9. stopnja: doktorat.

Smer izobrazbe

Smeri izobrazbe so ponavadi vezane na določeno znanost, vedo ali stroko. Navedimo nekaj različnih smeri izobrazbe na tehničnih področjih:

- računalniška,
- tehnična,
- naravoslovna,
- elektro,
- organizacijska,
- druge smeri.

Poklic

Poklic je smiselno zaokrožena celota znanj, sposobnosti, spretnosti in navad, ki posamezniku omogočajo, da opravlja dela in naloge določene zahtevnosti na nekem določenem področju dela neke dejavnosti. Če želimo opredeliti različne poklice v okviru določene dejavnosti, moramo najprej ugotoviti in določiti, kakšne so skupine del in nalog, ki jih v tej dejavnosti delavci opravljajo. Rezultat takega ugotavljanja oziroma z delitvijo dela povezanega preučevanja posameznih poklicev je seznam poklicev v neki dejavnosti. Pri tem moramo

upoštevati, da je poklic v primerjavi z deli, nalogami in opravili mnogo širši pojem ter da je tudi precej bolj statičen, kot so to dela in naloge, ki se danes zelo dinamično spreminjajo.

Vertikalni vidik klasifikacije poklicev v praksi običajno imenujemo kar klasifikacijska lestvica. Poklice, ki jih zahtevajo različne gospodarske in negospodarske dejavnosti, delimo danes glede na zahtevnost dela na osem kategorij, in sicer na:

- enostavna dela in naloge oziroma poklice,
- manj zahtevna dela in naloge oziroma poklice,
- srednje zahtevna dela in naloge oziroma poklice,
- zahtevna dela in naloge oziroma poklice,
- bolj zahtevna dela in naloge oziroma poklice,
- zelo zahtevna dela in naloge oziroma poklice,
- visoko zahtevna dela in naloge oziroma poklice,
- visoko zahtevna specializirana dela in naloge oziroma poklice.

Tuji jeziki: angleški, nemški, italijanski, francoski, ruski, ...

Določena delovna mesta v organizaciji zahtevajo od njihovih izvajalcev aktivno ali pasivno znanje enega ali več določenih tujih jezikov, predvsem to velja pri timih z visoko virtualnostjo.

Funkcionalna znanja

To so znanja, ki jih delavec z rednim šolanjem določene smeri ni mogel pridobiti, vendar jih za uspešno opravljanje dela v virtualnem timu potrebuje. Večja kot je virtualizacija skupine, večja je potreba po tehničnem znanju komuniciranja ter večja po samoorganiziranosti, iznajdljivosti in neodvisnosti.

Delovne izkušnje: do 5 let, do 10 let, do 20 let, do 30 let, ...

V modelu se bomo pod pojmom izkušnje omejili v glavnem na delovne izkušnje, čeprav so za uspešno opravljanje določenega dela lahko zelo pomembne tudi življenjske izkušnje. Vendar bomo predpostavili, da slednje lahko zadovoljivo ocenjujemo tudi s kriterijem starosti.

2.3.3 Ocena delovne uspešnosti

Delovna uspešnost posameznika je odvisna od njegove usposobljenosti in motivacije za delo. Na njegov delovni rezultat pa v pozitivnem ali negativnem smislu vplivajo njegove osebnostne lastnosti in delovna situacija (zlasti stil vodenja nadrejenega), v kateri delo opravlja.

Odvisno od področja in zahtevnosti dela oziroma posameznega delovnega mesta se uporabljajo za ocenjevanje delovne uspešnosti naslednji temeljni kriteriji:

- delovni rezultati (količina, kakovost, gospodarnost, doseganje ciljev),
- strokovnost (strokovna znanja, strokovna praksa, posebna znanja),
- odnos do dela (sodelovanje, samostojnost, iniciativnost, kreativnost ipd.),
- sposobnost vodenja.

V našem modelu smo uporabili naslednje kriterije:

Storilnost

Načeloma se ocenjujejo rezultati dela (količina, število, porabljen čas in podobno). Če je to možno le deloma, se oceni tudi prizadevanje (hitrost, fizični in/ali psihološki napor in podobno) in učinek.

Kljub temu je treba upoštevati, da se da ne glede na področje dela posameznega delavca vsaj deloma določen del nalog skoraj vedno tudi kvantificirati in na ta način tudi meriti. Take naloge naj po možnosti opredelita delavec in vodja skupaj, in sicer tako, da opredelita ustrezne kazalce - v obliki individualno ali standardno postavljenih ciljev - pred začetkom naslednjega ocenjevalnega obdobja. Po preteku obdobja se dosežene rezultate preverja in primerja s predpostavkami. Tudi spoštovanje rokov sodi v to področje.

Kakovost (kvaliteta)

Kakovost dela se ocenjuje pri pomožnih, proizvodnih in administrativno-tehničnih ter podobnih delih. V določenih primerih se zahtevana kakovost dela vnaprej predpiše (normira, standardizira). Zahtevana kakovost mora biti seveda ekonomsko upravičena, kar pomeni, da ne gre le za to, da bi bila kakovost čim višja, temveč da naj bi bila, odvisno pač od vrste dela, le tolikšna, kot je potrebno.

V primerih, ko zahtevana kakovost dela ni vnaprej natančno določena, jo skušamo oceniti na podlagi različnih podkriterijev, kot so: samostojnost, zanesljivost, prilagodljivost, potreben obseg nadzora in kontrole, interes za izpopolnjevanje in podobno.

Strokovnost ocenjujemo pri strokovnih in vodstvenih delih (pomeni nam torej merilo za kakovost dela). Kaže se v stopnji obvladovanja znanj in sposobnosti, potrebnih za uspešno opravljanje dela na določenem področju oziroma delovnem mestu. Pri izvajanju dela se strokovnost kaže predvsem v uporabi ustreznih metod in tehnik dela ter v sposobnosti zaznavanja medsebojnih povezav in odnosov pri reševanju kompleksnih problemov. Pravilna uporaba znanj in sposobnosti vodi k racionalnim načinom dela.

Strokovnost ocenjujemo torej predvsem na podlagi tega, kako posameznik obvladuje za njegovo delovno področje relevantne delovne postopke in tehnike dela. Tudi pri ocenjevanju strokovnosti si lahko pomagamo s podobnimi podkriteriji, kot smo jih navedli pri kakovosti dela. Dober, čeprav kvalitativen, kriterij za ocenjevanje strokovnosti so tudi vnaprej postavljeni cilji.

Odnos do dela

Pripravljenost za delo, angažiranost, pridnost, samostojnost, pripravljenost za prevzemanje odgovornosti in podobni pojmi opredeljujejo delavčev splošni odnos do dela oziroma njegovih delovnih obveznosti. Zlasti pomembno merilo je stopnja identifikacije posameznika z zaupanimi mu nalogami.

Čeprav pozitiven odnos do delovnih obveznosti pričakujemo od vsakega delavca in so zato določeni kriteriji za ocenjevanje univerzalni, pa so po drugi strani določeni kriteriji glede na vrsto in področje dela specifični. Tako je na primer lahko generalno merilo za odnos do dela stopnja angažiranosti posameznika pri izvajanju delovnih nalog, medtem ko sta upoštevanje predpisanih delovnih postopkov na eni strani ali pa fleksibilnost na drugi specifična kriterija. Določeno delo zahteva natančno upoštevanje predpisanih postopkov, drugo pa nasprotno terja iznajdljivost in prilagodljivost (iskanje novih rešitev).

Odnos do strank

Na določenih delovnih mestih imajo delavci tudi opravka z najrazličnejšimi strankami. Zato dela na teh delovnih mestih zahtevajo tudi ustrezen odnos delavca do strank. Od takih delavcev se zahteva ustrežljiv odnos do strank, korektnost, prijaznost, strpnost in podobno. Posebno pomembna pri vsem tem je delavčeva osebna urejenost in kultura komunikacije.

Odnos do sodelavcev

Eden od pogojev za uspešno delo so dobri medsebojni odnosi med zaposlenimi v organizaciji na vseh relacijah (delavec - sodelavec - nadrejeni - podrejeni). Pripravljenost za sodelovanje med sodelavci, ki so med seboj pri delu odvisni in povezani, kar naj prispeva k doseganju skupnih ciljev organizacije, lahko ocenjujemo na podlagi univerzalnih lastnosti, kot so: pripravljenost za medsebojno pomoč, poštenost, korektnost, sprejemanje kritike, toleranca, kolegialnost, pripravljenost za skupinsko delo in podobno.

Pri tem ne smemo pozabiti na upoštevanje pravil, kompetenc in pristojnosti. Te so lahko predpisane (normativni akti, organizacijski predpisi, varnostni predpisi ipd.) ali dogovorjene (interna delitev dela, razpored sestankov ipd.). Skratka, ne gre samo za spoštovanje napisanih, temveč tudi dogovorjenih pravil obnašanja.

Gospodarnost

S kriterijem gospodarnosti ocenjujemo predvsem ekonomično uporabo materialnih in kadrovskih virov. Pri tem pod gospodarnostjo razumemo namensko, skrbno, upravičeno in smotrno, skratka optimalno izrabo razpoložljivih virov in sredstev.

Pri delavcih na med seboj oddaljenih delovnih mestih je merilo gospodarnosti lahko tudi stopnja izkoristka razpoložljive infrastrukture organizacije (npr. informacijski sistem, managerski informacijski sistem ipd.), delovnih sredstev (npr. če imajo ti kadri osebne računalnike, naj jih tudi uporabljajo), finančnih sredstev (npr. učinkovitost naložb) in kadrov, ki so jim podrejeni.

Generalno merilo gospodarnosti je zavest o potrebi po stalnem zmanjševanju stroškov, ki naj bo cenjena vrednota, prisotna pri vseh zaposlenih.

Vodenje

Vodenje pomeni osebni stik in vplivanje na posameznika ter skupine, da bi delovali za skupne cilje. Pričakovani način delovanja in obnašanja sodelavcev vodja usmerja s pomočjo strokovne, osebnostne in pozicijske avtoritete.

Strokovnost vodenja lahko ocenjujemo predvsem na podlagi kvalitete vodstvenih odločitev na področjih, kot so: postavljanje ciljev, planiranje, organizacija, kontrola in nadzor ter podobno. Predvsem se mora vodja zavedati prednosti virtualizacije tima in te docela izkoristiti.

Pozicijsko pristojnost uresničuje vodja predvsem s sankcijami. Pri tem gre za zagotavljanje delovne discipline in za uresničevanje predpisanih in/ali dogovorjenih pravil delovnega obnašanja sodelavcev. Pozicijskih pristojnosti vodja ne sme prekoračevati niti zanemarjati.

Zdravstveno stanje

Delavci morajo biti za opravljanje določenega dela tudi zdravstveno sposobni in ustrezni. Glede na delo, ki ga opravljajo, morajo imeti ustrezne senzorne sposobnosti, mentalne, mehanične in motorične spretnosti.

Pri senzornih sposobnostih mislimo predvsem na vid in njegovo občutljivost za različne globine, opažanje globine in reliefnosti, razlikovanje barv, diferencialno občutljivost za svetlobo in barvne nianse, širino vidnega polja.

Za opravljanje številnih poslov in delovnih nalog je nujna posebna občutljivost vida, zato se pri predhodnem preverjanju delovnih sposobnosti kandidata uporabljajo tudi posebni aparati, specialno konstruirani za ta namen. Za opravljanje mnogih poslov in nalog je zelo pomemben tudi sluh.

Za opravljanje določenih del so potrebne sposobnosti hitre in natančne identifikacije oblik oziroma opažanja majhnih razlik med oblikami, potem sposobnosti zamišljanja in predstavljanja razporeda ter odnosa površin in predmetov v dve ali tri prostorske dimenzije, sposobnosti hitrega in natančnega izvajanja preprostih številnih operacij, verbalne sposobnosti, sposobnosti zgovornosti, inteligenca.

Za druga dela so zopet potrebne druge sposobnosti. Med mehanične sposobnosti štejemo sposobnosti razumevanja mehaničnih principov in reševanja tehnično-praktičnih problemov. Te sposobnosti so potrebne za posle in delovne naloge, ki se opravljajo na strojih, instrumentih in delih ali celotnih tehničnih napravah, kjer je uspešnost dela odvisna od razumevanja principov njihovega funkcioniranja.

2.3.4 Tehnični kriteriji

Virtualne time se praviloma uporablja v programerskih hišah, zato tudi kriterije glede na tehnično znanje izberemo iz področja računalniškega programiranja. Kljub temu da je programiranje zelo širok pojem, se ga da razdeliti v naslednje skupine: programski jeziki, programska orodja, baze podatkov, spletne tehnologije in administracija sistemov.

Znanje programiranja

Mnogo proizvajalcev s svojimi tehnologijami na trgu pomeni tudi mnogo programskih jezikov. Pri sodelovanju na različnih projektih je potrebno tudi različno znanje le-teh. Navedeni so najbolj razširjeni programski jeziki:

- C++,
- C#,
- Visual basic,
- Visual basic for .NET,
- Delphi,
- Java script,
- Java.

Poznavanje programerskih orodij

Različna programerska orodja omogočajo fleksibilno delo in izvajanje simulacij v različnih pogojih in s tem iskanje optimalne rešitve. Glede na potrebe projekta se izbere najprimernejše orodje. Navedena so trenutno najbolj razširjena programerska orodja:

- Visual studio,
- Net Beans,
- Eclipse,
- Oracle forms,
- Oracle designer,
- Microsoft Visio,
- NHibernate,
- Nunit.

Poznavanje baz podatkov

Podatkovne baze so se pojavile zaradi potrebe po hitrem dostopu do informacij, saj hramba podatkov iz preteklosti omogoča premišljeno odločanje o prihodnosti. Tak namen vodi današnja podjetja, da ustvarjajo zbirke podatkov o povpraševanju, strankah, vremenskih razmerah in sploh vsem, kar ima možnost vplivanja na bodoče poslovanje. Dandanes so aplikacije, ki ne bi uporabljale podatkovne baze, redkost, zato so različna znanja podatkovnih baz pomembna:

- database administration,
- database programming,
- database design,
- SQL,
- XML,
- UML.

Poznavanje spletnih tehnologij

Spletne aplikacije so v današnjem času interneta in integraciji spletnih storitev vse bolj uveljavljene. Spletne tehnologije so zelo širok pojem in zato so glede na naravo projekta potrebna samo določena znanja. Navedena so različna področja v sklopu spletnih tehnologij:

- ASP.NET,
- AJAX,
- PHP,
- web services,
- web servers (IIS, Apache,...),
- HTML,
- CSS and themes,
- Web design.

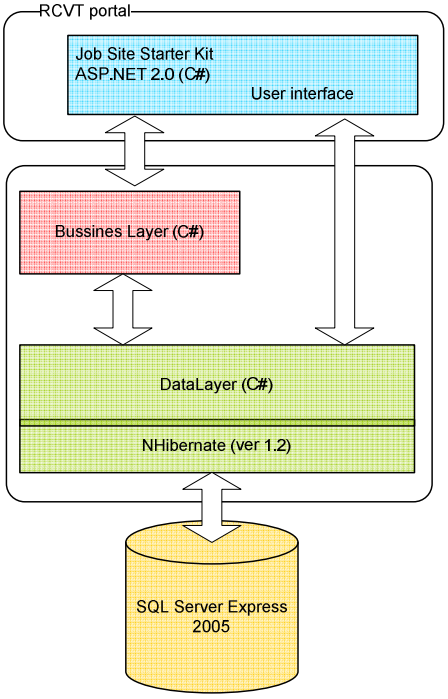
Administracija

Ocenjuje se administracija sistema Windows, in sicer administracija samih sistemov in pa administracija omeržij z operacijskim sistemom Windows. Ocenjuje se tudi administracija sistema in omrežij v sistemu Linux.

3 SISTEM ZA VODENJE BORZE DELA

3.1 Arhitektura portala

Portal je zgranjen v Microsoftovi .NET tehnologiji, uporabljeno je razvojno okolje Visual Studio 2008 Express in ogrodje .NET Framework 3.5. Primarni programski jezik je C#, za uporabniški vmesnik je poleg tega uporabljen še HTML, JavaScript in CSS. Aplikacija je izvedena kot spletna ASP.NET aplikacija s trinivojsko arhitekturo (Slika 3-1) [15]. Prvi nivo je uporabniški vmesnik (User Interface, Presentation Layer), katerega ogrodje sloni na spletni predlogi *ASP.NET Starter Kid*: Jobs Site Starter Kid. Prevzeta je zgradba uporabniškega vmesnika, osnovno ogrodje za prijavo, teme in menuji. Uporabniški vmesnik potrebuje za delovanje spletni strežnik *Internet Information Services*.



SLIKA 3-1: ARHITEKTURA REŠITVE

Nivo poslovne logike (Bussines Layer, Logic Layer) je knjižnica, ki vsebuje poslovno logiko aplikacije. Ta nivo vsebuje programsko logiko za podporo funkcijam portalnih procesov. Podatkovnemu nivoju posreduje zahteve po podatkih, rezultate teh poizvedb pa v primerni obliki posreduje uporabniškemu vmesniku. Hkrati skrbi za upoštevanje poslovnih pravil, kontrolo podatkov pred pošiljanjem v podatkovno bazo in podobno.

Podatkovni nivo (Data Access Layer) je knjižnica, ki oskrbuje sistem s podatki. Skrbi za vzpostavitev povezave do baze, izvedbo poizvedb, izvedbo operacij nad podatki in podobno. S podatki oskrbuje knjižnico poslovne logike in direktno uporabniški vmesnik. Podatke iz baze podatkov zajema s pomočjo NHibernate knjižnice (podrobnosti v poglavju 3.2.3). Ta knjižnica omogoča preslikavo med podatki v bazi in objekti, ki predstavljajo entitete.

Za podatkovno bazo se uporabi SQL Server 2005 Express, ki je brezplačen podatkovni strežnik, opisan v poglavju 3.2.2. Ker se za vmesnik do podatkovnega nivoja uporablja NHibernate se lahko za podatkovno bazo katerokoli drugo, ki jo NHibernate podpira.

3.2 Uporabljena tehnologija

3.2.1 ASP.NET Spletne aplikacije

ASP.NET je Microsoftovo ogrodje za spletno tehnologijo [1]. Z njim se na enostaven način izdeluje dinamične spletne strani, spletne aplikacije in spletne storitve. Samo ogrodje je del .NET tehnologije in je naslednik Microsoftove tehnologije aktivnih strani (ASP).

ASP.NET sestoji iz različnih tehnologij: spletne strani, spletne storitve, strežniške kontrole in HTML. Vse te tehnologije skupaj nam omogočajo, da izdelujemo robustne, vzdržljive spletne aplikacije. Spletne strani, ki vsebujejo različne spletne kontrole, se uporabljajo za uporabniški vmesnik. S spletnimi storitvami lahko izdelujemo distribuirane aplikacije.

ASP.NET spletne strani so izdelane s pomočjo aspx strani. Aspx je stran, ki vsebuje statično HTML ali XHTML kodo, kot tudi markup-e, ki definirajo spletne kontrole ali uporabniške spletne kotrole. V te kontrole se da vso zahtevano statično in dinamično vsebino spletne strani. Dinamično kodo, ki se izvršuje na strežniku, se doda na stran v blok <%-- dinamična koda --%>, kar je zelo podobno kot pri nekaterih ostalih spletnih razvojnih okoljih (PHP, JSP in ASP). Vendar tako vključevanje dinamične kode v spletno stran ni priporočeno s strani Microsofta, razen pri povezovanju komponent s podatki. Priporočena praksa je uporaba code-behind modela, ki loči kodo v posebno datoteko ali pa v za to posebej definiran prostor. Ko se uporablja ta model programiranja, programer piše kodo glede na dogodke, ki se izvajajo na spletni strani (stran se je naložila, uporabnik je pritisnil gumb,...).

ASP.NET uporablja naslednjo tehniko izvajanja: začetna aspx predloga se prevede v začetno kodo, ki zgradi drevo kontrol. Začetna koda je kombinirana z uporabniško napisano kodo in rezultati razreda, ki so značilni za to stran. Zahtevki za stran so obdelani skozi več korakov.

Najprej se skozi začetni korak inicializira razred strani in izvede začetna koda. Ta skreira začetno drevo kontrol, ki se nato spreminja s strani metod na strani. Ker je vsaka kontrola v drevesu predstavljena z instanco razreda, lahko koda spreminja strukturo drevesa kot tudi lastnosti z metodami posameznega objekta. Na koncu se skozi korak renderiranja strani renderira vsak vozel posebej. Rezultat tega je HTML koda, ki je nato poslana klientu. Ko je ta proces končan, se instanca strani odstrani, z njim pa tudi začetno drevo.

ASP.NET ima posebej rezervirano strukturo področij, kjer se nahajajo različne vrste datotek. Posebno rezervirana področja so:

App_Browsers: vsebuje datoteke, ki se uporabljajo glede na izbiro internet brskljalnika.

App_Code: na tem področju se nahaja samo koda. ASP.NET strežnik avtomatično prevede vse datoteke (na tem področju in vseh podpodročjih) v knjižnico, ki je dostopna vsaki strani v projektu. App_Code se ponavadi uporablja za kodo, ki jo abstrahira podatkovni nivo, kodo modulov in implementacijo spletnih storitev.

App_Data: na tem področju se nahajajo datoteke, ki so podlaga podatkovnim bazam, kot na primer Accessove mdb datoteke ali SQL Server mdf datoteke. To področje je ponavadi edino, ki ima dovoljenje pisanja za aplikacijo.

App_LocalResources: vsebuje datoteke za lokalizacijo, in sicer za vsako stran posebej. Datoteke so poimenovane ime_strani.aspx.sl-SL.resx za slovensko verzijo ime_strani.aspx strani. Ko se spremeni kultura uporabniškega vmesnika, ASP.NET avtomatično poišče to datoteko in jo uporabi za lokalizacijo strani.

App_GlobalResources: vsebuje datoteke resx za lokalizacijo, ki so enotne za vse strani v projektu. V teh datotekah se tipično shranjujejo lokalizirana sporočila, ki so uporabljena na več straneh.

App_Themes: vsebuje datoteke, ki vsebujejo različne stilne oblike (teme) projekta.

App_WebReferences: vsebuje WSDL datoteke, ki so reference do zunanjih spletnih storitev.

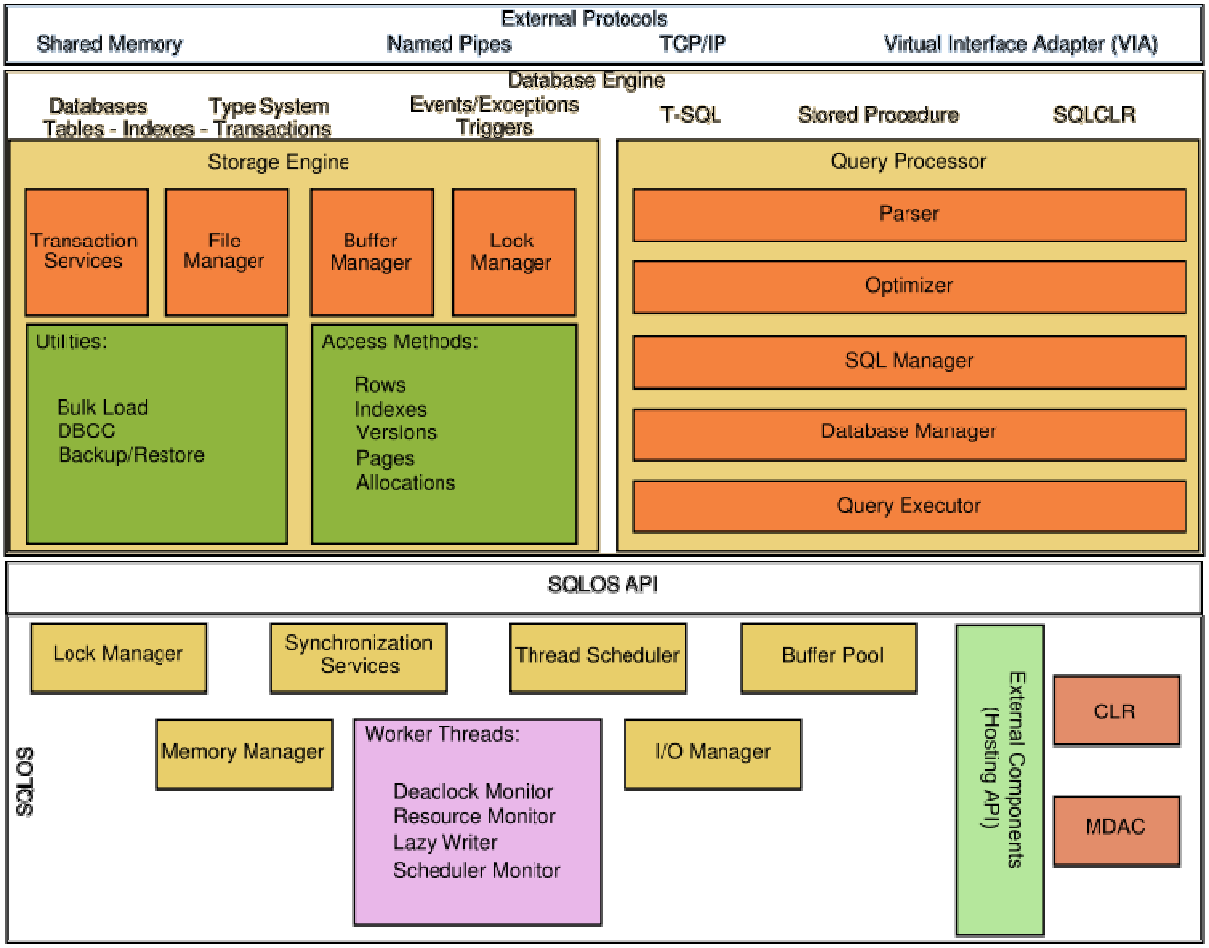
Bin: vsebuje prevedene knjižnice (.dll datoteke) kontrol, komponent in druge kode, ki jo referenciramo iz projekta. Vsak razred, ki je prisoten na Bin področju, je avtomatično referenciran v projektu.

ASP.NET skuša izboljšati učinkovitost, tako, da prevede vso strežniško kodo v dll knjižnice, ki se nahajajo na spletnem strežniku. To prevajanje se izvši avtomatično ob prvem zagonu spletne aplikacije, ob naslednjih zagonih prevajanje ni več potrebno.

3.2.2 SQL Strežnik

Microsoft SQL Server je relacijska baza podatkov (RDBMS), njen glavni poizvedovalni jezik je Transact-SQL, ki ustreza ANSI/ISO standardu SQL-a [19].

Arhitektura SQL Strežnika je v grobem razdeljena na tri dele (Slika 3-2): *SQLOS*, ki implementira osnovne storitve za delovanje podatkovnega strežnika; to so nadzorovanje niti, upravljanje spomina in vhodno/izhodnih operacij. *Relational engine* implementira komponente podatkovne baze, kot so podpora za bazo, tabele, poizvedbe in bazne procedure. *Protocol layer* implementira zunanji vmesnik do strežnika SQL Server.



SLIKA 3-2: ARHITEKTURA SQL STREŽNIKA

Glavna funkcionalnost baze je zajem podatkov iz baze. Poizvedbe, ki deklarirajo željen zajem podatkov, morajo biti za SQL Server napisane v T-SQL sintaksi. Vsaka poizvedba se analizira in naredi se plan poizvedbe. Vsako poizvedbo se lahko izvede na več možnih načinov, s pomočjo plana poizvedbe pa SQL Server skuša najti način, ki bi pomenil najkrajši čas za zajem podatkov.

3.2.3 NHibernate

Veliko kontrol v .NET knjižnici ima možnost enostavnega povezovanja direktno na podatke v podatkovni bazi. To pomaga programerjem, da na hiter način naredijo enostavne aplikacije. Vendar, ko začne aplikacija vsebinsko rasti, se začne dodajati poslovna logika in stvari postanejo bolj kompliciranе. Podatkovna baza postane večja in kompleksnejša. Z vsako spremembo v aplikaciji je ogroženo obsoječe delovanje, saj sta podatkovna baza in aplikacija močno povezani. Spremembe v aplikaciji lahko vplivajo na podatke in obratno.

ORM pomaga preprečiti opisano situacijo. Ogrodje za preslikavo med podatkovno bazo in poslovnimi objekti v aplikaciji služi namenu, da ob dostopu ali shranjevanju podatkov enostavno zajemamo objekte oziroma shranjujemo objekte, ki predstavljajo zahtevane podatke. Glavna ideja je narediti abstrakten dostop do baze v popolnoma ločenem nivoju in podatke obdelovati skozi poslovne objekte in ne direktno v podatkovno bazo.

Največja prednost uporabe ORMja je v organizaciji aplikacije v diskretne dele oziroma v rahlo sklopljene dele. Sprememba v strukturi podatkovne baze pomeni samo spremembo poslovnih objektov, ne vpliva pa na ostale dele aplikacije. Enostavnejši je razvoj velikih aplikacij, kjer teče razvoj na več nivojih. Projektne skupine enostavneje razdelijo razvojna področja. Del programerjev lahko razvija podatkovni model in poslovne objekte. Drugi del pa dela s poslovnimi objekti in jih komponira z drugo funkcionalnostjo aplikacije.

NHibernate je odprto kodno ORM ogrodje za .NET okolje [16]. Že dolgo let Hibernate prevladuje kot ORM rešitev v svetu Java programiranja. Hibernate je preizkušena tehnologija in je trenutno standardni pristop za velike aplikacije v ogromno računalniških rešitvah po celem svetu. Prav zato je bilo samo vprašanje časa, kdaj bo Hibernate prenešen tudi na .NET okolje. Tako je nastal NHibernate, ki pa se od Hibernate rešitve ne razlikuje veliko.

NHibernate implementira ORM koncepte skozi ogrodje, ki preslikuje relacijske podatkovne table v poslovne objekte s pomočje standardnega XML formata. NHibernate je neodvisen od baze podatkov, kar pomeni, da so preslikovalne datoteke neodvisne od baze. NHibernate s preslikovanjem podpira veliko vrsto objektno orientiranih konceptov, kot so dedovanje, polimorfizmi in tudi .NET zbirke vključno z generiki. Obvladovanje neštetih poizvedb v bazo in klicev baznih procedur v velikih poslovnih aplikacijah je lahko zelo potratno in težavno delo za vsakega programerja. NHibernate omogoči, da programerju ni potrebno skrbeti za večino problemov skladnosti podatkov s podatkovno bazo.

3.3 Funkcionalnost portala

Glavni namen portala za management človeških virov je ustvariti veliko množico potencialnih kandidatov, spremljanje njihove uspešnosti in sposobnosti. Za te potrebe vodenja evidence ponudnikov in prosilcev dela na področju razvoja informacijskih rešitev smo izdelali informacijsko podporo, ki bo omogočala:

- registracijo prosilcev dela in vnos njihovih podatkov;
- registracijo ponudnikov dela;
- iskanje po bazi prosilcev in izbiro najprimernejših kandidatov za konkreten projekt ali delo;
- vodenje komunikacije med ponudniki in prosilci dela.

#	Funkcija	Kratko ime	Opis
F1	Prijava	Prijava	Omogoča prijavo v sistem. Za prijavo se uporablja uporabniško ime in geslo.
F2	Registracija	Registracija	Omogoča, da se ponudniki in prosilci dela, ki bi želeli sodelovati v okviru borze dela, registrirajo kot uporabniki sistema.
F3	Definiranje povezave	Relacije	Iskalcem dela omogoča, da podajo podatke o tem, s katerimi drugimi iskalci, ki so registrirani v borzi dela, se poznajo.
F4	Pregled znancev	Znanci	Omogoča pregledovanje povezav med iskalci dela.
F5	Zamrznitev članstva	Zamrznitev	Omogoča, da se določeno članstvo zamrzne. Razlogi za zamrznitev so lahko različni; npr. neaktivnost.
F6	Pošiljanje e-maila	E-mail	Deluje kot poštni strežnik in omogoča pošiljanje elektronske pošte.
F7	Potrjevanje članstva	Potrjevanje članstva	Prosilcu omogoča, da potrdi članstvo v borzi dela, potem ko se je registriral. Potrditev poteka prek elektronske pošte.
F8	Potrjevanje ponudbe	Potrjevanje ponudbe	Prosilcu omogoča, da potrdi pripravljenost za sodelovanje na določenem projektu ali nalogi, potem ko je bil izbran s strani ponudnika in mu je bila ponudba poslana. Potrditev poteka prek elektronske pošte.

F9	Potrjevanje znanstva	Potrjevanje_znanstva	Prosilcu omogoča, da potrdi znanstvo, ki ga je evidentiral nek drugi prosilec dela. Potrditev poteka prek elektronske pošte.
F10	Generiranje zaslonskih mask	Generiranje GUI	Omogoča avtomatsko izdelavo zaslonskih mask za potrebe vnosa in pregledovanja podatkov o prosilcih dela.
F11	Vnos in urejanje podatkov o članih in ocenah	Ocene članov	Omogoča vnos in urejanje podatkov ter ocen o posameznih prosilcih dela, ki so registrirani v bazi borze dela.
F12	Ocenjevanje dela na projektu	Ocene projektov	Omogoča vnos ocen za posameznega virtualnega člana, ki sodeluje na nekem projektu.
F13	Vzdrževanje podatkov o projektih	Projekti	Omogoča vnos, brisanje in spreminjanje podatkov o projektih.
F14	Iskanje prosilcev	Iskanje prosilcev	Omogoča iskanje za nek projekt oziroma nalogo najprimernejših kandidatov med prosilci dela.
F15	Generiranje ponudbe za sodelovanje	Ponudbe	Omogoča avtomatsko izdelavo ponudbe za sodelovanje za izbrane kandidate. Ponudbe se posredujejo po elektronski pošti.

Funkcije so namenjene naslednjim akterjem (Slika 3-3):

- **prosilcem**: z oznako prosilec označujemo osebe, ki so registrirane v bazi borze dela kot prosilci dela;
- **ponudnikom**: z oznako ponudnik označujemo vse pravne subjekte, ki so registrirani v bazi borze dela kot ponudniki dela;
- **poštni strežnik**: ima funkcijo pošiljanja elektronske pošte prosilcem dela.

SLIKA 3-3: PRIMER UPORABE ZA KADROVSKI PORTAL

F1 Prijava

Člani vpišejo za prijavo svoj elektronski naslov in geslo. Ko se preveri pravilnost gesla (polje Password iz tabele JobSeekers), se vstopi v portal, ki ima omogočene vse funkcije, ki ustrezajo prosilcem zaposlitve.

Ponudniki se prav tako vpišejo z elektronskim naslovom in navedejo geslo. Preveri se pravilnost gesla (polje Password iz tabele Employers) in po verifikaciji se vstopi v portal, ki ima vse omogočene funkcije za ponudnika zaposlitve.

F2 Registracija

Člani se registrirajo v sistem tako, da vpišejo e-mail naslov, ki služi kot enolična identifikacija člana. Kot obvezni vnosi so še ime in priimek ter geslo. Prav tako se ob registraciji vnese v tabelo JobSeekers rang člana = 1, v polje Created se vpiše datum in čas, ko je bil član vnešen. Član naj se ob vpisu vnese kot nepotrjen (Confirmed='N'), ker se mu ob vnosu pošlje elektronsko pošto s povezavo, ki potrdi vnos. V primeru, da član ne potrdi svojega vpisa, se mu vpis izbriše. Član naj bo aktiven (Active = 'Y').

Ponudniki zaposlitve se prav tako registrirajo v sistem enolično glede na elektronski naslov. Zapisi ponudnika gredo v tabelo Employers. Vnese se ime, naslov in kontaktne podatke ponudnika. Za verifikacijo pa se vnese tudi geslo. V polje Created se vpiše datum in čas, ko je bil ponudnik vnešen. Ponudnik naj se ob vpisu vnese kot nepotrjen (Confirmed='N'), ker se mu ob vnosu pošlje elektronsko pošto s povezavo, ki potrdi vnos. V primeru, da ponudnik ne potrdi svojega vpisa, se mu vpis izbriše. Ponudnik naj bo aktiven (Active = 'Y').

F3 Definiranje povezave

Člani, ki se med seboj poznajo, se lahko v sistemu med seboj povežejo. To storijo tako, da označijo osebe, ki jih poznajo. Drugi osebi se nato pošlje e-mail, v katerem se jo naproša, da potrdi povezavo.

Član izbere drugega člana, ki ga pozna. V seznamu so prikazani zgolj tisti člani, ki jih do sedaj še ni definiral kot povezave. Prikazati je potrebno priimek in ime (»Name« in »Surname« v tabeli »JobSeekers«), pri čemer se osebi med seboj še ne poznata (ne obstaja povezava v tabeli »Relations«). Član nato v polje, kako se poznata, vnese tudi opis vrste poznanstva (glej prototip zaslonske maske).

Član pritisne »poveži«. Uporabnika povežemo tako, da v tabelo »Relations« vnesemo novo vrstico: *JobSeeker_ID* = id uporabnika člana, ki definira povezavo (session spremenljivka »user-id«), *JobSeeker_Relation_ID* = id člana, s katerim se želimo povezati, *RelationOffered* = »T«, *RelationAccepted* = »F«, *Description* = opis, kako se poznata (zaslonska maska).

Članu, s katerim se želimo povezati, je potrebno poslati sporočilo, v katerem ga prosimo, naj potrdi oziroma zavrže poznanstvo.

F4 Pregled znancev

Uporabnik sistema lahko pregleduje svoje znance. Znanci so tisti člani, s katerimi se je v okviru sistema povezal.

Identifikacija znancev, povezanih s trenutnim članom: povezavo vidimo v tabeli »Relations«. Trenutno prijavljeni uporabnik mora biti enak »JobSeeker_ID« ali »Related_JobSeeker_ID« v tabeli »Relations«, poleg tega pa mora biti vrednost atributa »RelationAccepted« enaka »Y«. Prikažemo »Name« in »Surname« iz tabele »Job Seekers« ter »RelationAcceptedDate« in »Description« iz tabele »Relations«.

F5 Zamrznitev članstva

Uporabnik lahko zamrzne članstvo, kar pomeni, da sistema nekaj časa ne bo uporabljal. S tem najavi, da trenutno ne želi sodelovati na novih projektih.

Uporabniku prikažemo njegovo trenutno stanje. Atribut, ki nakazuje, ali je član aktiven, se imenuje »Active« (vrednost »Y« za aktiven in »N« za neaktiven) in se nahaja v tabeli »JobSeekers« za člane oziroma v tabeli »Employers« za zaposlovalce. V kolikor je stanje aktivno, uporabniku prikažemo gumb »Zamrzni članstvo«. S klikom na ta gumb spremenimo stanje atributa »Active« iz »Y« na »N«. Nato skočimo nazaj na stanje »Odmrzni članstvo«.

V kolikor je stanje neaktivno, uporabniku prikažemo gumb »Odmrzni članstvo«. S klikom na ta gumb spremenimo stanje atributa »Active« iz »N« na »Y«. Nato skočimo nazaj na točko »Zamrzni članstvo«.

F6 Pošiljanje e-maila

Primer uporabe pošiljanja elektronske pošte omogoča pošiljanje tovrstne pošte številnim drugim primerom uporabe.

S strani druge forme dobimo naslednje parametre: naslov sporočila, e-mail prejemnika in samo sporočilo. Oblikujemo sporočilo. Pošljemo sporočilo preko SMTP strežnika.

F7 Potrjevanje članstva

Ko se uporabnik registrira v sistem, mu pošljemo e-mail, v katerem uporabnika pozovemo, da potrdi članstvo. Uporabnik potrdi članstvo tako, da v e-mailu klikne na ustrezno povezavo.

Uporabnik dobi e-mail in klikne na povezavo do strani za potrjevanje članstva.

Stran kot parametre dobi naslednje podatke:

- role pomeni vlogo uporabnika in lahko zavzame dve vrednosti »S« (job seeker) ali »E« (employer);
- id uporabnika je numerična vrednost, ki enolično identificira uporabnika in je enaka ID-ju v podatkovni bazi;
- action pove, ali je uporabnik potrdil ali zavrnil članstvo in lahko zajema vrednosti »Yes« (potrjuje članstvo) ali »No« (ne potrjuje).

Če uporabnik potrdi članstvo, to zapišemo v podatkovno bazo tako, da v kolikor gre za člane (role=S), v tabeli »JobSeeker« v vrstico z ustreznim ID-jem v atribut »Confirmed« zapišemo vrednost »Y«, če pa gre za zaposlovalce (role=E), v tabeli »Employers« v vrstico z ustreznim ID-jem v atribut »Confirmed« zapišemo vrednost »Y«. Na uporabniški vmesnik izpišemo »Aktivacija uspešna.« in povezavo do prijave.

Če uporabnik ne potrdi članstva, uporabnika izbrišemo iz podatkovne baze. V kolikor gre za člane (role=S), izbrišemo vrstico z ustreznim IDjem iz tabele »JobSeeker«, če pa gre za zaposlovalce (role=E), izbrišemo vrstico z ustreznim IDjem iz tabele »Employer«. Na uporabniški vmesnik izpišemo »User <ime> <priimek> removed from the database.«.

F8 Potrjevanje ponudbe

Prosilcu se omogoča, da potrdi pripravljenost za sodelovanje na določenem projektu ali nalogi, potem ko je bil izbran s strani ponudnika in mu je bila ponudba poslana. Potrditev poteka preko elektronske pošte. Uporabnik dobi e-mail in klikne na povezavo do strani za potrjevanje ponudbe za delo.

Stran kot parametre dobi naslednje podatke:

- id ponudbe je numerična vrednost, ki je enaka ID-ju tabele »Offer« v podatkovni bazi;
- action pove, ali je uporabnik sprejel ali zavrnil ponudbo in lahko zajema vrednosti »Yes« (sprejme ponudbo) ali »No« (ne sprejme).

Če uporabnik sprejme ponudbo, to zapišemo v podatkovno bazo tako, da v tabelo »Offer« v vrstico, ki ima Offer_ID enak id-ju ponudbe, podanemu preko parametra, v polje »OfferAcceptedDate« zapišemo trenutni datum in čas. Na uporabniški vmesnik izpišemo »Offer accepted.«.

Če uporabnik zavrne ponudbo, ne naredimo ničesar. Na uporabniški vmesnik izpišemo »Offer declined.«.

F9 Potrjevanje znanstva

Ko en član označi drugega člana kot znanca, drugemu članu pošljemo e-mail, v katerem ga prosimo, da potrdi to navedbo. Uporabnik potrdi znanstvo tako, da v e-mailu klikne na ustrezno povezavo. Uporabnik dobi e-mail in klikne na povezavo do strani za potrjevanje članstva.

Stran dobi naslednje parametre:

- id relation je numerična vrednost, ki enolično določa relacijo med članom, ki je definiral povezavo, in članom, za katerega je definiral, da ga pozna;
- action je tekstovno polje, ki vsebuje informacijo o tem, ali član potrjuje poznanstvo (»Yes« - potrjuje, »No« - zavrača).

Če uporabnik potrdi poznanstvo, to zapišemo v podatkovno bazo tako, da v tabeli »Relations« v vrstico z ustreznima ID-jem, v atribut »RelationAccepted«, zapišemo vrednost »Y«, v atribut »RelationAcceptedDate« pa današnji datum. Na uporabniški vmesnik izpišemo »Link between users created succesfully.«.

Če uporabnik zavrne poznanstvo, ne naredimo ničesar. Na uporabniški vmesnik izpišemo »Link between users declined.«.

F10 Generiranje zaslonskih mask

Primer uporabe Generiranje zaslonskih mask vključuje skupne funkcije za dinamično izdelavo zaslonskih mask za prikaz, vnos in ažuriranje podatkov o članih. Podatki za generiranje mask so zapisani v tabelah v podatkovni bazi. Podatke o atributih je mogoče dodajati in spreminjati v okviru drugih primerov uporabe.

S strani drugega primera uporabe pride zahteva za generiranje zaslonske maske. V okviru zahteve so podani naslednji parametri: tip zaslonske maske; tipi atributov, ki naj se prikažejo (tabela AttributeTypes); JobSeeker_ID (atribut iz tabele JobSeeker).

Zahteva se glede na tip obravnava na tri različne načine. Gre lahko za zahtevo za izdelavo zaslonske maske za:

- prikaz obstoječih vrednosti (generira se zaslonska maska z vsemi ustreznimi polji, ta pa se napolnijo s podatki);
- vnos novih vrednosti (generira se zaslonska maska z vsemi ustreznimi polji, ki so uporabniku na voljo za vnos novih vrednosti);
- spremembo obstoječih vrednosti (generira se zaslonska maska z vsemi ustreznimi polji, ta pa se napolnijo s podatki; maska omogoča spreminjanje teh podatkov).

Podtok Prikaz obstoječih vrednosti

Sistem v skladu s pravili in vsebino posameznih tabel zgradi zaslonsko masko za prikaz. Pri tem uporabi podatke iz tabel Attributes, AttributeValues, AttributeGroups in AttributeTypes, v primeru lookup atributa pa še iz tabele LookUpValues. Ob gradnji maske hkrati tudi bere vrednosti iz tabele CV in z njimi napolni masko za prikaz. Podatke iz CV izbere na podlagi JobSeeker_ID (atribut iz tabele JobSeeker). Pri prikazu se upošteva tudi Rang. V primeru, da je Rang v tabeli JobSeeker nižji od zahtevanega v tabeli Attributes, se atribut ne prikaže.

Podtok Vnos novih vrednosti

Sistem v skladu s pravili in vsebino posameznih tabel zgradi zaslonsko masko za vnos novih vrednosti, v kateri so vnosna polja, ki so prazna. Pri tem uporabi podatke iz tabel Attributes, AttributeValues, AttributeGroups in AttributeTypes, v primeru lookup atributa pa še iz tabele LookUpValues. Pri prikazu se upošteva tudi Rang. V primeru, da je Rang v tabeli JobSeeker nižji od zahtevanega v tabeli Attributes, se atribut ne prikaže.

Sistem prikaže tipko pošlji. Uporabnik vnese vrednosti in pritisne tipko pošlji. Sistem rezultat pošlje v obliki skupne podatkovne strukture (objekt, DataSet, ipd. – strukturo si zamislite sami in jo dokumentirajte v opisu vmesnika) funkciji, ki je primer uporabe sprožila.

Podtok Sprememba obstoječih vrednosti

Sistem v skladu s pravili in vsebino posameznih tabel zgradi zaslonsko masko za vnos novih vrednosti. Pri tem uporabi podatke iz tabel Attributes, AttributeValues, AttributeGroups in AttributeTypes, v primeru lookup atributa pa še iz tabele LookUpValues. Ob gradnji maske hkrati tudi bere vrednosti iz tabele CV in z njimi napolni masko za vnos. Podatke iz CV izbere na podlagi JobSeeker_ID (iz tabele JobSeeker). Pri prikazu se upošteva tudi Rang. V primeru, da je Rang v tabeli JobSeeker nižji od zahtevanega v tabeli Attributes, se atribut ne prikaže.

Sistem prikaže tipko pošlji. Uporabnik vnese vrednosti in pritisne tipko pošlji. Sistem rezultat pošlje v obliki skupne podatkovne strukture funkciji, ki je primer uporabe sprožila.

Alternativni tokovi

V primeru, da v tabeli CV manjka podatek, ki ga je potrebno prikazati, se prikaže prazno polje. V primeru, da uporabnik v polje ne vnese vrednosti, se v podatkovni strukturi za to polje vrne vrednost null.

Generiranje zaslonskih mask

Pri generiranju maske upoštevate naslednje:

1. Prikaz atiributov združujete v zaključene skupine glede na njihovo članstvo v določeni skupini. Podatke o tem preberete iz tabele AttributeGroups. Vsi atributi, ki spadajo v skupno grupo, so prikazani na skupnem delu spletne strani (npr. podobmočje na strani ipd.). Vrstni red atributov v grupi določa polje DisplayOrder.
2. Nadalje združujete skupine atributov glede na njihovo skupno pripadnost, ki je razvidna iz table AttributeTypes. Vsi atributi oz. skupine atributov istega »tipa« spadajo na skupno enoto za prikaz (npr. skupno spletno stran, jeziček na stani, območje na strani ipd.). Vrstni red grup na strani določa polje DisplayOrder.
3. Pri polnjenju atributov tipa lookup (combobox) se vrednosti preberejo iz tabele LookUpValues.

Slika 3-4 pojasnjuje, kako približno naj izgledajo različni elementi generirane strani. S slike je razvidna osnovna funkcionalnost posameznih elementov na generirani strani. Dokončno oblikovanje (barve, velikost, font ipd.) je določeno s strani predloge CSS.

SLIKA 3-4: ZASLONSKA MASKA

F11 Vnos in urejanje podatkov o članih in ocenah

Primer uporabe Vnos in urejanje podatkov o članih in ocenah omogoča vnos in ažuriranje podatkov ter ocen o posameznih članih - prosilcih dela, ki so registrirani v bazi borze dela. Primer uporabe neposredno ne vključuje prezentacijskega nivoja, saj za prikaz uporablja komponento K10_Generiranje_GUI.

Ponudnik ali član na osnovni maski pritisne na gumb za *Vnos/urejanje podatkov člana*. Sistem ugotovi, ali za člana že obstajajo podatki, in:

- v primeru, da podatki o članu že obstajajo, sproži primer uporabe generiranje zaslonskih mask (komponenta K10) za Spremembo obstoječih vrednosti (podtok za urejanje) oz.
- v primeru, da podatkov o članu še ni, sproži primer uporabe generiranje zaslonskih mask (komponenta K10) za Vnos novih vrednosti (podtok za vnos).

Podatke, ki jih vrne primer uporabe generiranje zaslonskih mask, zapiše v tabelo CV skladno s pravili vnosa podatkov v CV (glej opis tabele CV). Pomembno je, da se podatki nikoli ne brišejo, ampak se samo spreminja njihova veljavnost (atribut Valid v CV). Če ponudnik ali član vnos podatkov predčasno zaključi, se podatki ne vpišejo.

F12 Ocenjevanje dela na projektu

Primer uporabe Ocenjevanje dela na projektu se nanaša na ocenjevanje kakovosti dela in sodelovanja virtualnih članov. Primer uporabe je na voljo delodajalcem in se uporablja po zaključku nekega opravila ali projekta.

Delodajalec na osnovni zaslonski maski pritisne gumb Ocenjevanje dela na projektu. Delodajalec iz seznama projektov izbere tistega, s katerim želi vnašati ocene dela virtualnih članov. Prikažejo se samo projekti delodajalca, ki je prijavljen v sistem. Kdo pa je ta, se ugotovi iz podatkov, ki se vodijo v seji (*Session*). Kot identifikator delodajalca (*ID_Employer*) se vedno uporablja elektronski poštni naslov. Delodajalec iz seznama izbere virtualnega člana, ki ga želi oceniti (prikažejo se samo imena in priimki virtualnih članov, ki opravljajo delo na izbranem projektu – glej tabelo *Offers*, stolpec *Project_ID).*

Delodajalec iz seznama izbere vrsto opravila, ki ga je izbrani virtualni član opravljal v okviru izbranega projekta (glej tabelo *WorkTypes*). Delodajalec v vnosno polje *Ocena dela* poda opisno oceno kakovosti dela in sodelovanja izbranega virtualnega člana na izbranem projektu. Vnosna polja potrdi s pritiskom na gumb *Shrani*. Sistem shrani podatke v podatkovno bazo (tabela *Offer*).

F13 Vzdrževanje podatkov o projektih

Primer uporabe Vzdrževanje podatkov o projektih je namenjen ponudnikom za vnos, brisanje in spreminjanje podatkov o njegovih projektih.

Na osnovni maski ima ponudnik razpredelnico projektov, ki se nahajajo v tabeli Projects. V razpredelnici vedno vidi samo svoje projekte, glede na Employer_ID. Projekte lahko ureja ali pa jih briše. Projekta se ne more izbrisati, v kolikor je bila na njem sprejeta kaka ponudba v tabeli Offers. Nov projekt se doda s klikom na gumb Nov projekt. Nov projekt vsebuje kratko in dolgo ime, opis projekta, začetni in končni datum realizacije projekta. Ko se projekt shrani, se mora v Employer_ID zapisati, kateri ponudnik je nosilec projekta.

F14 Iskanje kandidatov

Iskanje kandidatov se izvaja glede na potrebe projekta, ki se nahajajo v tabeli Projects. Glede na hierarhijo kriterijev se za vsako vozlišče generirajo matrike povezav z možnostjo vnosa pomembnosti parametrov v tem vozlišču (poglavje 5.2 in priloga A). Vrednosti pomembnosti parametrov se vnaša v tabelo ProjectAttributes. Na podlagi te tabele se izvede izračun in iskanje po članih s pomočjo algoritma, opisanega v poglavju 5.5.

F15 Generiranje ponudbe za sodelovanje

Primer uporabe Generiranje ponudbe za sodelovanje se sproži, ko ponudnik izmed kandidatov, ki ustrezajo iskalnim pogojem, izbere tiste, ki jim želi posredovati ponudbo za sodelovanje (glej tudi primer uporabe Iskanje delojemalcev).

Ponudba za sodelovanje se posreduje po elektronski pošti. S strani primera uporabe Iskanje delojemalcev pride zahteva za generiranje ponudbe za sodelovanje. V okviru zahteve sta podana naslednja parametra: seznam vrednosti *JobSeeker_ID*, ki pove identifikacijske oznake izbranih kandidatov (glej tabelo *JobSeekers*), ter vrednost *Project_ID*, ki pove identifikacijsko oznako projekta, za katerega se pripravlja ponudba za sodelovanje (glej tabelo *Projects*).

- Sistem sestavi tekst sporočila.
- Sistem pošlje elektronska sporočila. Za pošiljanje elektronskega sporočila uporablja primer uporabe Pošiljanje e-maila.

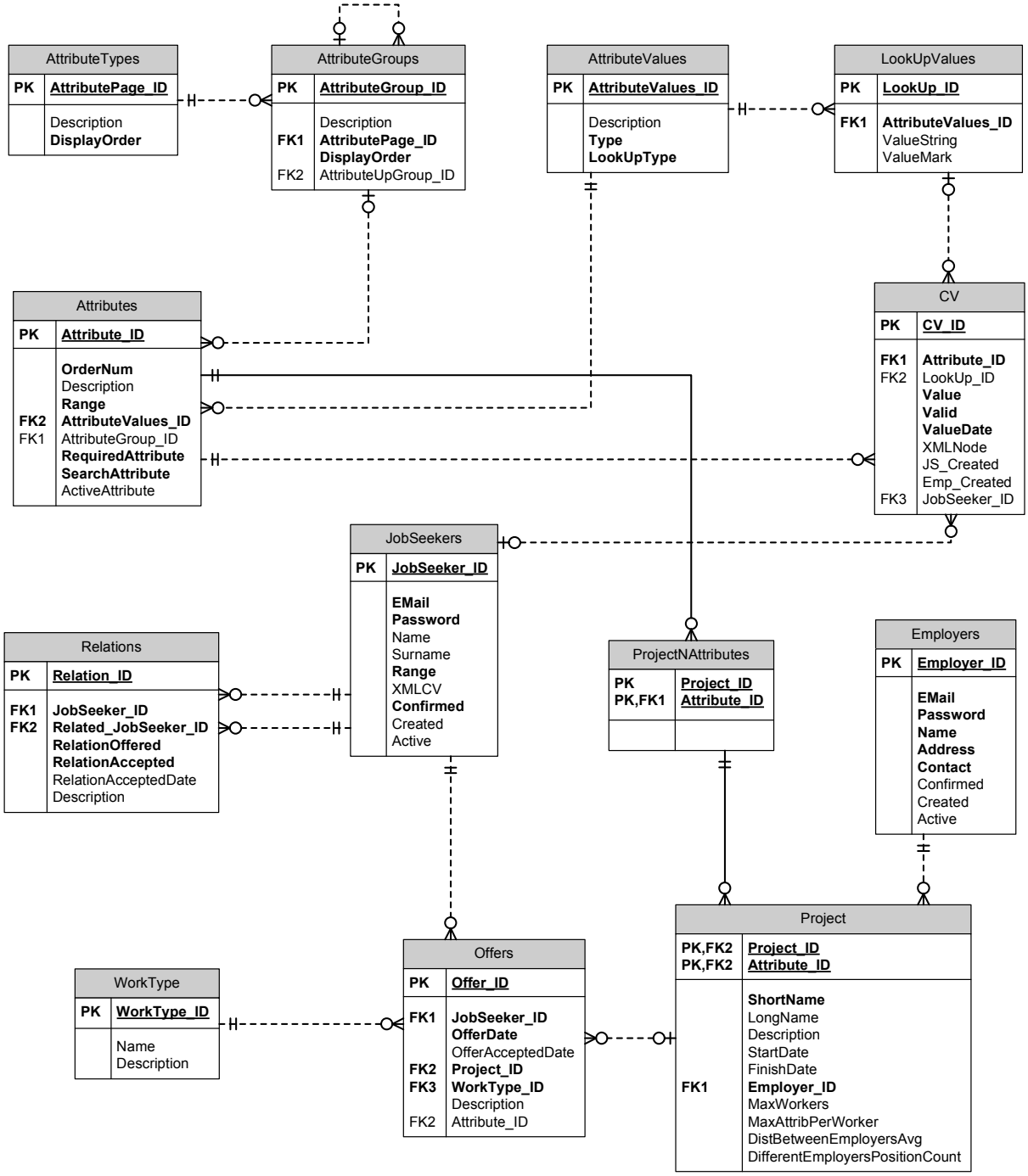
Če v okviru zahteve po izvedbi primer uporabe ne prejme ustreznih parametrov, izpiše sporočilo o napaki.

3.4 Entitetni model

Za potrebe sistema je na voljo **podatkovna baza**, ki je prikazana na sliki 3-5 in opisana v nadaljevanju.

#	Tabela	Opis
T1	Employeers	Vsebuje podatke o ponudnikih.
T2	Projects	Vsebuje podatke o projektih, za katere ponudniki iščejo pomoč pri prosilcih dela.
T3	Offers	Vsebuje podatke o konkretnih nalogah, ki jih posamezni prosilci dela sprejmejo na projektih.
T4	WorkTypes	Vsebuje podatke o vrstah dela, ki jih lahko nek prosilec dela izvaja.
T5	JobSeekers	Vsebuje podatke o prosilcih dela.
T6	Relations	Vsebuje podatke o povezavah med posameznimi prosilci dela.
T7	CV	Vsebuje podatke o profilu posameznega iskalca dela.
T8	Attributes	Vsebuje podatke o lastnostih, ki jih beležimo za vsakega prosilca dela.
T9	AttributeValues	Vsebuje podatke o podatkovnemu tipu atributa. Če so vrednosti, ki jih atribut lahko zavzame, točno določene, so te navedene v tabeli LookUpValues.
T10	LookUpValues	Vsebuje podatke o vrednostih, ki jih lahko zavzamejo »lookup« atributi.
T11	AttributeGroups	Vsebuje podatke o tem, kako so atributi grupirani v skupine.
T12	AttributeTypes	Vsebuje podatke o razvrstitvi atributov po področjih na svojo stran na pregledih.
T13	ProjectNAttributes	Določi, kateri atributi morajo biti ocenjeni pri vseh prosilcih ob iskanju tima za določen projekt.

TABELA 1: TABELE PODATKOVNE BAZE



SLIKA 3-5: ENTITETNI MODEL ZA KADROVSKI PORTAL

T1 PONUDNIKI:

Employer_ID	**int**
EMail	varchar(100)
Password	varchar(30)
Name	varchar(50)
Address	varchar(1000)
Contact	varchar(100)
Confirmed	char(1)
Created	datetime
Active	char(1)

Podatki o ponudnikih: glavni identifikator člana je e-mail, ki je enoličen. Določen ponudnik ne more imeti istega e-maila kot kak član.
Potrditev ponudnika mora biti s strani administratorja portala, da ne pride do zlorab statusa ponudnika. Če je ponudnik nekativen, potem se obravnava, kot da ga ni v bazi.

T2 PROJEKTI:

Project_ID	**int**
ShortName	varchar(20)
LongName	varchar(100)
Description	varchar(500)
StartDate	datetime
FinishDate	datetime
Employer_ID	int
MaxWorkers	int
MaxAttribPerWorker	int
DistBetweenEmployersAvg	int
DifferentEmployersPositionCount	int

Projektu določimo kratko in dolgo ime ter njegov naziv. Opišemo trajanje projekta od do. Ponudnik, ki prijavlja projekt, se zapiše v Employer_ID.
Za potrebe iskanja imamo še polja MaxWorkers, kjer povemo, kolikšno bo število članov v timu, ter MaxAttribPerWorker, kjer povemo, koliko različnim neobveznim atributom ustreza član. Atributa DistBetweenEmployersAvg in DifferentEmployersPositionCount služita za določanje virtualizacije, DifferentEmployersPositionCount pa za število članov na različnih lokacijah in DistBetweenEmployersAvg za povprečno razdaljo med člani.

T3 KANDIDATI ZA PONUDBE:

Offer_ID	**int**
Project_ID	int
WorkType_ID	int
JobSeeker_ID	int
OfferDate	datetime
OfferAccepted	char(1)
OfferAcceptedDate	datetime
Description	varchar(100)

To je tabela ponudb kandidatom. Ko ponudnik, za določen projekt (Project_ID) in določeno delo (WotkType_ID), išče med zadetki kandidatov, lahko označi posamezne, če se hočejo odzivati na njegovo ponudbo. V primeru, da kandidat potrdi ponudbo (OfferAccepted=«D«), lahko ponudnik spreminja njegove podatke. Vendar lahko spreminja samo atribute, ki še nimajo vrednosti.

T4 VRSTE DELA:

WorkType_ID	**int**
Name	varchar(30)
Description	varchar(250)

Vrsto dela, ki je potrebno pri projektih, opišemo z nazivom in opisom dela.

T5 ČLANI:

JobSeeker_ID	**int**
EMail	varchar(100)
Password	varchar(30)
Name	varchar(50)
Surname	varchar(30)
Range	decimal(2, 0)
Confirmed	char(1)
Created	datetime
Active	char(1)

Podatki o članih. Glavni identifikator člana je e-mail, ki je enoličen.
Rang člana se določa s strani ponudnika.
Ob registraciji člana je član nepotrjen. Član mora potrditi registracijo preko e-maila. V primeru, da ne potrdi registracije v določenem času, se registracija zbriše.
Če je član nekativen, potem se obravnava, kot da ga ni v bazi.

T6 ZNANSTVA:

Relation_ID	**int**
JobSeeker_ID	int
Related_JobSeeker_ID	int
RelationOffered	datetime
RelationAccepted	char(1)
RelationAcceptedDate	datetime
Description	varchar(100)

To je tabela znanstev. Ko je znanstvo vpisano, se vpiše datum vpisa znanstva (RelationOffered).
Vsako znanstvo mora biti potrjeno s strani člana, kateremu je bilo znanstvo predlagano.
Dokler znanstvo ni potrjeno (RelationAccepted), je brezpredmetno.
V primeru, da je znanstvo potrjeno, lahko spreminjamo podatke svojemu znancu. Vendar lahko spreminjamo samo atribute, ki še nimajo vrednosti.

T7 PODATKI O ČLANU:

CV_ID	**int**
JobSeeker_ID	int
Attribute_ID	varchar(50)
LookUp_ID	int
Value	varchar(100)
Valid	char(1)
ValueDate	datetime
JS_Created	int
Emp_Created	int

Za vsak atribut vpišemo njegovo vrednost glede na člana. Če je atribut, ki ima LookUpType = »D«, potem se v LookUp_ID vpiše ID LookUp vrednosti. V Value se vseeno zapiše še LookUp vrednost.
Spremembe vrednosti atributov se izražajo z novim vnosom, ki ima nov datum (ValueDate). Stara vrednost se naredi kot neveljavna (Valid = N).
Zbris vrednosti pomeni samo postavitev atributa na neveljavnega.
V primeru, da je atribut zahtevan, mora biti vsaj en zapis veljaven.
V primeru, da atribut ni multiatribut, je lahko veljaven samo en zapis njegove vrednosti na člana.
Če vrednost atributa vnese kdo drug, ki ni član, torej znanec ali ponudnik, se zabeleži, kdo je vrednost vnesel v polja JS_Created, če gre za znanca, oziroma v Emp_Created, če gre za ponudnika.

T8 ATRIBUTI:

Attribute_ID	**varchar(50)**
OrderNum	int
Description	varchar(255)
Range	decimal(2, 0)
AttributeValues_ID	varchar(10)
AttributeGroup_ID	varchar(10)
RequiredAttribute	char(1)
SearchAttribute	char(1)
ActiveAttribute	char(1)

Šifrant atributov vsebuje atribute, ki jim lahko določimo vrednost glede na člana.
Vsak atribut ima opis in določimo katerega tipa je njegova vrednost.
Rang atributa nam pove vsaj, kakšen rang mora imeti član, če ga hoče urejati. V primeru, da je rang = 0, ga lahko ureja že takoj na začetku.
Atribute lahko grupiramo. Grupa je, ko so atributi med seboj odvisni. Na primer, znanje tujega jezika (atribut tuj jezik) in ocena tega jezika (atribut ocena) morata biti v isti grupi.
Required polje nam pove, ali je vnos tega atributa obvezen (Y) ali pa je neomogočen (N) vnos vrednosti s strani člana. Tako vrednost lahko vnaša samo ponudnik.
Iskalni atribut (SearchAttribute) je atribut, po katerem se bo izvajalo iskanje. Če atribut ni iskalni, se ga samo prikazuje na pregledu člana.
Če atribut ni aktiven, zanj ni mogoče vnašati vrednosti.

T9 VREDNOSTI ATRIBUTOV:

AttributeValues_ID	**varchar(10)**
Description	varchar(50)
Type	varchar(10)
LookUpType	char(1)

Vrednost atributa nam pove, katere vrste je atribut. Glede na vrsto atributa določimo na strani vnosno predlogo.
Tip vrednosti atributa (type) nam pove, ali je vrednost tekstovna, številčna ali datumska.
Določimo lahko tudi, ali imamo za atribut že domene vrednosti (LookUpType = Y), ki jih lahko uporabnik izbere na vnosni maski.

T10 DOMENE VREDNOSTI ATRIBUTOV:

LookUp_ID	**int**
AttributeValues_ID	varchar(10)
ValueString	varchar(500)
ValueMark	int

To so domene vrednosti za atribute, ki so določeni, da morajo biti njihove vrednosti izbrane iz domen. Vrednost izbremo glede na tip vrednosti atributa. V primeru, da je tekstovna, potem je domena vrednosti v stolpcu ValueString, če pa je številčna, pa v ValueMark.

T11 GRUPE ATRIBUTOV:

AttributeGroup_ID	**varchar(10)**
Description	varchar(50)
AttributePage_ID	varchar(10)
DisplayOrder	int
AttributeUpGroup_ID	varchar(10)

Grupa atributa nam opisno (Description) pove skupino atributa.
Grupi atributa določimo, na kateri strani (AttributePage_ID) se bo prikazovala in pa vrstni red grupe na strani s poljem DisplayOrder.
Nadgrupa za to grupo se določi v AttributeUpGroup_ID. To uporabimo pri hierarhično urejenih grupah.

T12 TIPI ATRIBUTOV:

AttributePage_ID	**varchar(10)**
Description	varchar(50)
DisplayOrder	int

Tip atributa nam opisno (Description) pove skupino atributa.
Glede na tip atributa jih postavimo ločeno vsakega na svojo stran. Vrstni red strani določimo s poljem DisplayOrder.

T13 POTREBNI ATRIBUTI:

Project_ID	**int**
Attribute_ID	**varchar(10)**

Tukaj določimo, kateri atributi bodo pri iskanju ocenjeni (izbrani) za vse prosilce dela glede na tim, ki se izbira za projekt.

4 LINEARNO PROGRAMIRANJE IN VEČPARAMETRSKI ODLOČITVENI PROBLEMI

4.1 Metodologija operacijskih raziskav

Terminološke definicije operacijskih raziskav so različne [2]:

- Operacijsko raziskovanje je znanstveni pristop k pripravi in izbiri optimalnih rešitev.
- Operacijsko raziskovanje je kvantitativna veda, ki pomaga upravljavcem pri sprejemanju dobrih rešitev.
- Operacijsko raziskovanje pomeni uporabo matematičnih metod pri modeliranju sistemov in analizi njihovih značilnosti za podporo (poslovnemu) odločanju.

Povzamemo jih lahko kot množico aplikativnih kvantitativnih metod za kakovostnejše odločanje pri "usmerjanju delovanja poslovnih in drugih sistemov".

Gre za kvantitativen način pristopanja odločevalca k reševanju danih problemov, ki ga poznamo tudi pod imenom kvantitativno modeliranje. Iz splošne teorije in raziskav odločanja vemo, da je kvantitativen način odločanja objektiven, nepristranski, definiran in ponovljiv. Kvalitativen, po drugi strani, pa bazira predvsem na intuiciji, subjektivnosti in izkušnjah iz podobnih preteklih situacij. Pri kompleksnih, pomembnih in nepoznanih problemih, pri katerih ima odločevalec čas za analizo in odločitev, bi naj managerji uporabljali kvantitativen-znanstven pristop in se naj ne bi odločali "po občutku". Kljub temu na tem mestu ni odveč dodati, da pri poslovnih problemih ustrezna mera kvalitativnosti in subjektivnosti vseeno sodi k odločanju – sposoben odločevalec ne sme slepo slediti rezultatom reševanja modela, ampak jih mora kritično ovrednotiti in se šele nato odločiti.

Metodologija operacijskih raziskav narekuje potek reševanja problema in odločanja (torej potek kvantitativega modeliranja v managementu) v fazah, ki so v kratkem predstavljene v preostanku tega razdelka [2]:

Definiranje problema

Prva faza odločevalcu narekuje formuliranje problema. Gre verjetno za najobčutljivejšo in najtežjo fazo kvantitativnega modeliranja, saj je od nje odvisno, kako relevantne zaključke bomo lahko potegnili iz rešitev.

Izgradnja modela

V tej fazi je, na podlagi definicije problema, zgrajen matematični model, ki je poenostavljena in (pogosto) idealizirana abstrakcija realnosti. Tak model na matematični način (npr. v obliki enačb) opisuje stanje sistema, pogoje, ki jih postavlja okolje sistema, in odnose med spremenljivkami modela. Definiramo torej odnose med odločitvenimi spremenljivkami, neobvladljivimi vhodi in izhodi sistema; določimo tudi sistem pogojev, ki določajo možno rešitev. Povedano z drugimi besedami, določiti je potrebno, o čem se odločamo, kako naše (različne) odločitve vplivajo na zastavljeni cilj (ki je izražen v obliki tako imenovane namenske funkcije), kakšen je cilj in kakšne so objektivne danosti (pogoji), ki nas omejujejo pri doseganju cilja.

Zagotovitev vhodnih podatkov

Ko imamo model zgrajen, je potrebno zagotoviti vhodne podatke, ki jih bo model uporabil za reševanje problema. Ta faza je zelo pomembna, kajti če naredimo še tako realen model, nam pri odločanju ne bo mogel pomagati, če mu ne bomo zmožni posredovati vhodnih podatkov. Po principu Garbage In-Garbage Out nam napačni podatki dajo tudi napačne rezultate. Dostikrat se zgodi, da popolnoma ustreznih in našemu modelu prirejenih podatkov ne dobimo. Razlogi so lahko različni: podatkov sploh ne zajemamo, so (pre)stari ali pa jih zajemamo v drugačni obliki. V takšnih primerih imamo več možnih rešitev, od tega da pripravimo izvedene podatke (iz obstoječih), uporabimo (ustrezne) ocene ali pa celo priredimo zgrajen matematični model. Tudi če so model in podatki že "kompatibilni", je slednje običajno potrebno še prečistiti, da ne bi morebitne napake pokvarile rezultatov našega modeliranja.

Reševanje modela

Ko je matematičen model formuliran in so podatki zbrani, sledi faza izgradnje procedure za reševanje modela. Običajno gre preprosto za apliciranje določenega standardnega algoritma-metode operacijskih raziskav oz. preprosto za uporabo enega izmed obstoječih računalniških orodij.

Tako lahko npr. probleme linearnega programiranja hitro rešimo s klasičnim "pisarniškim" orodjem, preglednico z vgrajenimi analitičnimi pripomočki (npr. Reševalec v orodju Microsoft® Excel). Poleg matematičnega programiranja (kamor spada linearno in nelinearno programiranje), ki je ena izmed najbolj uporabljanih tehnik kvantitativnega modeliranja, se pogosto uporabljajo še: simulacije, mrežno modeliranje, teorija vrst, odločitvena drevesa, celoštevilsko programiranje, dinamično programiranje, markovska analiza in ciljno programiranje. Včasih pa se zgodi, da za optimalno rešitev določenega problema še ne obstaja

procedura reševanja. V tem primeru lahko, če cost-benefit analiza to upravičí, poskušajo s pomočjo specialistov proceduro izumiti, ali, kar se dogaja pogosteje, poskušajo dobiti vsaj zadovoljive rešitve. Po Paretovem zakonu lahko namreč že z 20% napora (ali npr. stroškov) dosežemo 80% učinek. Tega se zavedajo tudi managerji, ki se "dostikrat odločijo za zadovoljivo rešitev, namesto da bi brezupno iskali optimalno".

Analiza rešitev

Ko dobimo rešitev, jo najprej formalno preizkusimo, če ustreza podanim omejitvam. Predvsem v nekaj začetnih poskusih reševanja je potrebno zgrajeni matematični model iterativno prilagajati in korigirati.

Sledi vsebinska analiza rezultatov, v kateri pogostokrat primerjamo več možnih rešitev med seboj in predvsem v povezavi z okolico, ki v model ni bila vključena. Možno je, da npr. "optimalna" rešitev s stališča namenske funkcije ni sprejemljiva za podjetje kot celoto, zato je potrebno poiskati kakšno podoptimalno rešitev, ki pa bo ustreznejša z vidika celote.

Aplikacija modela in nadziranje implementacije

Ko je rešitev izbrana, sledi njena implementacija, katere uspešnost je potrebno analizirati. Če gre za model, ki se večkrat ali celo avtomatizirano uporablja za določene izračune, mu moramo vgraditi tudi test, ki bo opozoril na spremembo strukture modela. Lahko se namreč zgodi, da nenadoma uporabljamo dober kvantitativni model za reševanje problema A pri reševanju problema B, kjer pa ni več ustrezen.

4.2 Formulacija linearnega programa

Linearno programiranje je ena najpopularnejših tehnik operacijskih raziskav, ki se ukvarja s tako imenovano optimizacijo z omejitvami. Povedano na kratko, gre za alociranje omejenih resursov na med seboj konkurenčne aktivnosti na najboljši (optimalen) način.

Linearni program (v nadaljevanju LP) je optimizacijski problem, za katerega velja [7]:

1. Za LP poskušamo maksimizirati ali minimizirati linearno funkcijo odločitvenih spremenljivk. Funkciji, ki jo maksimiramo ali minimiramo, pravimo *namenska oz. kriterijska oz. ciljna funkcija.*

2. Vrednosti odločitvenih spremenljivk morajo zadovoljiti nabor omejitev oz. pogojev. Vsak pogoj mora biti podan v obliki linearne enačbe ali linearne neenačbe.

3. Z vsako odločitveno spremenljivko je povezan pogoj nenegativnosti. Za vsako spremenljivko x_i je določeno, da mora biti x_i nenegativna ($x_i \geq 0$), ali – redkeje – da spremenljivka x_i ni pogojena glede nenegativnosti.

Pri formulaciji LP moramo določiti naslednje komponente:

1. **Identificirati odločitvene spremenljivke.** Rečemo jim tudi neznanke ali neodvisne spremenljivke, ki popolnoma opisujejo odločitve, ki jih lahko sprejmemo.

2. **Identificirati omejitve** in jih zapisati v obliki linearnih enačb ali neenačb odločitvenih spremenljivk. Na primer pri odločanju o proizvodnem in/ali prodajnem asortimentu podjetja imamo tipično opraviti s proizvodnimi (ki jih določa npr. tehnologija, produkcijski faktorji…) in tržnimi omejitvami (koliko lahko prodamo na trgu). Dodajmo še, da ima vsaka omejitev štiri elemente: (1) konstanto na desni strani (b_i), ki podaja mejo za obravnavani pogoj; (2) algebraičen znak (≤, = ali ≥), ki označuje tip pogoja; (3) odločitvene spremenljivke, na katere se omejitev nanaša (x_i); in (4) vpliv enote odločitvene spremenljivke (a_i oz. c_i) na desno stran omejitve oz. namenske funkcije.

3. **Identificirati namensko funkcijo** in jo zapisati v obliki linearne funkcije odločitvenih spremenljivk. Pri LP namreč poskušamo maksimizirati ali minimizirati dano namensko funkcijo s spreminjanjem odločitvenih spremenljivk.

4. **Identificirati parametre** odločitvenih spremenljivk v naboru omejitev (a_i) in koeficiente namenske funkcije (c_i). Gre za fiksne vrednosti, ki določajo vpliv, ki ga ima enota odločitvene spremenljivke na namensko funkcijo in omejitev.

Iščemo torej maksimum funkcije

$$f(x) = c_1 * x_1 + c_2 * x_2 + \ldots + c_n * x_{1n} \qquad (1)$$

pri linearnih pogojih

$$\begin{aligned} a_{11} * x_1 + a_{12} * x_2 + \ldots + a_{1n} * x_n &\leq b_1 \\ a_{21} * x_1 + a_{22} * x_2 + \ldots + a_{2n} * x_n &\leq b_2 \\ a_{m1} * x_1 + a_{m2} * x_2 + \ldots + a_{mn} * x_n &\leq b_m \end{aligned} \qquad (2)$$

in pogojih nenegativnosti

$$x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0 \qquad (3)$$

pri čemer sta indeksa *m* in *n* poljubni naravni števili, koeficienti a_{ij} (i=1,…,m; j=1,…,n) in c_j (j=1,…,n) poljubna realna, b_i (i=1,…,m) pa poljubna nenegativna števila.

Temu zapisu pravimo tudi *standardna oblika zapisa LP*, ki jo lahko namesto v skalarni obliki zapišemo tudi v matrični obliki, in sicer takole:

Določiti je potrebno vektor X, ki zadošča pogoju nenegativnosti

$$X \geq 0 \qquad (4)$$

in matrični enačbi

$$AX = B \qquad (5)$$

tako, da ima namenska funkcija

$$f\ (x)\ =\ CX \qquad (6)$$

maksimum.
Pri tem so vektorji

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, B = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \qquad (7)$$

Pri določanju naštetih komponent LP (odločitvene spremenljivke, namenska funkcija, omejitve in parametri) oz. pri formulaciji LP moramo upoštevati naslednje predpostavke:
Linearnost – Vsaka odločitvena spremenljivka ima linearen vpliv v namenski funkciji in v omejitvah, v katerih nastopa.
Aditivnost – Leva stran omejitev oz. namenske funkcije je vsota vplivov vseh odločitvenih spremenljivk.
Zveznost – Vrednosti odločitvenih spremenljivk so lahko katerekoli vrednosti v določenem intervalu. Pri praktični uporabi rešitve lahko omejene resurse delimo na tako drobno, kot želimo.
Gotovost – V modelu vladajo vzročno-posledične zveze, ki so fiksno določene. Ko določimo vrednosti odločitvenih spremenljivk, lahko takoj izračunamo stanje sistema oz. vrednost namenske funkcije.
Nenegativnost - Pogoji nenegativnosti so "logična posledica" vsebine problemov, ki se jih lotevamo z metodami linearnega programiranja – odločitvene spremenljivke predstavljajo take spremenljivke, ki ne morejo zavzeti negativnih vrednosti. Npr. pri odločanju o količini proizvodov v prodajnem asortimentu ne moremo izbrati -10 proizvodov A in 5 proizvodov B; podobno pri investicijskem odločanju ne moremo kupiti 10 delnic A, 5 delnic B in -100 delnic C.

Pomembno je, da imamo te predpostavke v mislih še pred postavitvijo modela oz. takrat, ko o uporabi določene metode operacijskih raziskav za reševanje problema, na katerega smo naleteli, šele razmišljamo.

4.3 Reševanje LP

Najosnovnejša pojma, povezana z reševanjem linearnih programov, sta območje dopustnih rešitev in optimalna rešitev, to je tista (najboljša možna) rešitev, ki jo iščemo. Območje *dopustnih rešitev* je nabor vseh točk, ki zadoščajo vsem pogojem linearnega programa. *Optimalna rešitev* za maksimizacijski LP je tista točka območja dopustnih rešitev, ki ima največjo vrednost namenske funkcije. Podobno, optimalna rešitev za minimizacijski linearni program je tista točka območja dopustnih rešitev, ki ima najmanjšo vrednost namenske funkcije [2].

Gledano s stališča rešitev, spada vsak LP v eno izmed kategorij:
1. LP ima **eno optimalno** rešitev.
2. LP ima **alternativne oz. več optimalnih** rešitev. Namenska funkcija v dveh ali več ekstremnih točkah na območju dopustnih rešitev dosega optimalno vrednost. Ker to pomeni, da več različnih kombinacij odločitvenih spremenljivk daje enake (optimalne) rešitve, se

lahko odločevalec odloči za (poljubno) tisto kombinacijo, ki mu najbolj ustreza tudi z vidika drugih (širših) dejavnikov.
3. **Optimalna rešitev ni v območju dopustnih rešitev**. LP nima nobene možne rešitve, saj nobena od točk ni v območju dopustnih. V takem primeru je na primer potrebno:

- "Omehčati" kakšnega od pogojev s povečanjem obsega (redkih) resursov. Spremeni se vektor desnih strani omejitev (*b*).
- Zmanjšati "porabo" resursov in odpraviti ozka grla. Spremenijo se koeficienti matrike (*A*).

4. LP **ima neomejeno množico** rešitev. V tem primeru je mogoče vrednost namenske funkcije po absolutni vrednosti poljubno večati (maksimizacijski LP) oz. manjšati (minimazacijski LP). Do takšne situacije pride predvsem takrat, ko smo neustrezno formulirali model – če smo npr. pri maksimizacijskem LP pozabili dodati kakšno (≤) omejitev ali če smo pri minimizacijskem LP pozabili dodati pogoje nenegativnosti.

4.4 Odločanje

Odločanje je proces, v katerem moramo med več variantami (inačicami, alternativami, možnostmi) izbrati tisto, ki najbolj ustreza postavljenim ciljem oziroma zahtevam [8].

Po psihološki plati je primerjava več variant in izbira najustreznejših med njimi težko opravilo. Že posamezna varianta je običajno tako kompleksna, da do konca obremeni naš kratkoročni spomin. Seveda je medsebojna primerjava več takih variant še težje breme za naše umske sposobnosti. Z drugimi besedami, če kompleksnost variante razumemo s podajanjem različnih parametrov in če je teh res veliko, takih variant v glavi ni možno primerjati. V poštev pride tehnika strukturiranja, kjer se postopoma ukvarjamo s po nekaj parametri naenkrat. Poleg tega se vedno srečujemo tudi z neznanimi elementi. Del odločitve je oprt na predvidevanja. Ne vemo, kaj se bo v resnici pripetilo. Spet drugih elementov preprosto ne poznamo. Da bi jih spoznali, včasih ni ne časa in ne možnosti.

Tako dejansko stanje stvari velikokrat naslika problem odločitve v nemogoči luči, kar ima včasih za posledico odlaganje odločitve, dokler je to še mogoče. S tem izgubljamo čas, da bi se odločitvene situacije lotili sistematično in organizirano. Lahko se zgodi, da odločitev enostavno obidemo, prepustimo naključju ali pa se odločamo šele tedaj, ko smo res prisiljeni. S tem pa je odločanje vse prej kot tehtanje vseh komponent in končna izbira najustreznejše variante. Rezultat tega so nepremišljene in večinoma slabe odločitve.

Poleg izbire najboljše variante želimo včasih ožji izbor variant tudi rangirati. Pri tem so variante objekti, akcije, scenariji ali posledice enakega oziroma primerljivega tipa. Kadar na primer kupujemo nov avto, so variante avtomobili. Pri strateškem planiranju v podjetju so lahko variante različni razvojni ali investicijski scenariji. Kadar pa izbiramo najboljšega kandidata za neko delovno mesto, so variante ljudje.

Odločanje je običajno del splošnega reševanja problemov in nastopa kot pomembna mentalna aktivnost na praktično vseh področjih človekovega delovanja. Težavnost odločitvenih

problemov je zelo raznolika. Sega od enostavnih osebnih odločitev, ki so večinoma rutinske in se jih največkrat niti ne zavedamo, vse do težkih problemov skupinskega odločanja, na primer pri vodenju, upravljanju in planiranju v podjetjih, kadrovskem odločanju, medicinski diagnostiki in vrsti drugih področij. Najpomembnejši problemi, ki nastopajo pri težkih odločitvenih problemih, izvirajo iz:

- velikega števila dejavnikov, ki vplivajo na odločitev;
- številnih oziroma slabo definiranih ali poznanih variant;
- zahtevnega in pogosto nepopolnega poznavanja odločitvenega problema in ciljev odločitve;
- obstoja več skupin odločevalcev z nasprotujočimi si cilji;
- omejenega časa in drugih virov za izvedbo odločitvenega procesa.

Poleg tega se moramo zavedati, da na naše odločitve in dejanja bistveno vplivajo človekove omejitve miselnih sposobnosti. Velikokrat naletimo na razkorak med postopki, ki bi jim morali slediti, in tistimi, po katerih res delamo. Naša dejanja se nam zdijo premišljena in logična tudi takrat, ko niso. Večino takega ravnanja se da pojasniti z »lokalno« logiko, ki je osnovana na delni informaciji, ki jo tisti hip upošteva odločevalec. Ko kasneje opazimo očitno napako, pravimo, da smo pač spregledali nekaj pomembnega.

Iz tega sledi, da moramo proučevati odločanje kot sistematičen racionalen proces in ob tem upoštevati človekove omejitve, ki se pogosto kažejo v nehoteni iracionalnosti, to je v navidez brezciljnih dejanjih.

Vse navedeno o pojmu odločanja lahko zaključimo s formalno definicijo, da pri odločanju nastopa:

1. **Množica variant *A***: $a_1, a_2, a_3, \dots, a_m, \dots$

Odvisno od problema je *A* lahko končna ali pa tudi neskončna.

2. **Preferenčna relacija P**

Relacija *P* uredi množico *A* po zaželenosti, ustreznosti oziroma koristnosti. Racionalna odločitev pomeni izbiro tiste variante *a* iz *A*, ki je najbolj zaželena. V splošnem je takih variant lahko več. V primeru neskončne množice *A* pa v praksi ponavadi obravnavamo neko končno podmnožico.

V odločitveni praksi navadno skušamo vpeljati *funkcijo koristnosti* oziroma zaželenosti. Funkcija *v(a)* izmeri stopnjo zaželenosti variante *a*, tako da za vsak par *a*, *b* iz *A* velja:

$$a\ P\ b \Leftrightarrow v(a) > v(b) \qquad (8)$$

kjer *a P b* pomeni, da imamo varianto *a* rajši kot *b*.

Racionalna odločitev je potem izbira variante a_k, tako da je:

$$v(a_k) = \max\,(v(a): a \in A) \qquad (9)$$

Pri tem se srečujemo z mnogimi zanimivimi problemi. Osnovni je identifikacija relacije *P*, ki predstavlja preferenčno znanje, to je znanje, na osnovi katerega odločamo. Pove nam, kaj je bolj zaželeno, koristno, vredno v neki odločitveni situaciji. Relacija *P* pa je neposredno le redko tudi praktično uporabna. Skušamo jo zamenjati s funkcijo zaželenosti *v*. Pri tem naletimo na probleme merjenja. Gre za transformacijo »naravnega« sistema preferenc v neki »računsko« primernejši sistem. Pomembno pa je, da se s tem naše osnovno odločitveno znanje ne spremeni. Ostati mora enako, le izraženo na drug način, ki je primernejši za obravnavo.

4.5 Merjenje koristnosti in logika odločanja

Naloga racionalne teorije odločanja je identificirati informacijo, pomembno za odločitev, in določiti, kako to informacijo združiti tako, da pridemo do zaključka, to je običajno najboljše variante in s tem odločitve. Osnovni princip racionalnega odločanja je optimizacija, kar pomeni izbrati varianto z največjo vrednostjo.

Optimizacija deluje tako, da maksimiziramo dobre strani in minimiziramo slabe, kar je v principu zelo enostavno. Vendar v tej enostavni obliki nima prave praktične vrednosti. Ugotoviti moramo, kako človek v odločitveni situaciji ocenjuje dobre in slabe strani. Vidimo, da jih različni ljudje ocenjujejo različno. Psihološka vrednost, ki jo človek pripisuje neki varianti ali objektu, se imenuje koristnost oziroma zaželenost tega objekta. V osnovi gre za problem določanja mere koristnosti. Problem pa je, kako se mere uporabljajo v poteku odločanja.

Določanje koristnosti moremo razumeti kot merjenje koristnosti in ga obravnavati v okviru teorije merjenja, ki omogoča količinsko oceno koristnosti posameznih variant oziroma inačic za odločevalca. V terminologiji teorije merjenja gre za identifikacijo ustrezne merske lestvice. Ta mora izražati preferenčno znanje odločevalca. Določiti moramo lestvico, ki ustreza odločitvenemu problemu. Problem je torej, kako priti do ustrezne funkcije koristnosti. V skladu s teorijo merjenja moramo zagotoviti obstoj in enoličnost te funkcije. To pa pomeni, da moramo preverjati ustreznost aksiomov v izrekih. Tak primer je preverjanje tranzitivnosti. Določanje funkcije koristnosti s pomočjo preverjanja aksiomov imenujemo aksiomski pristop.

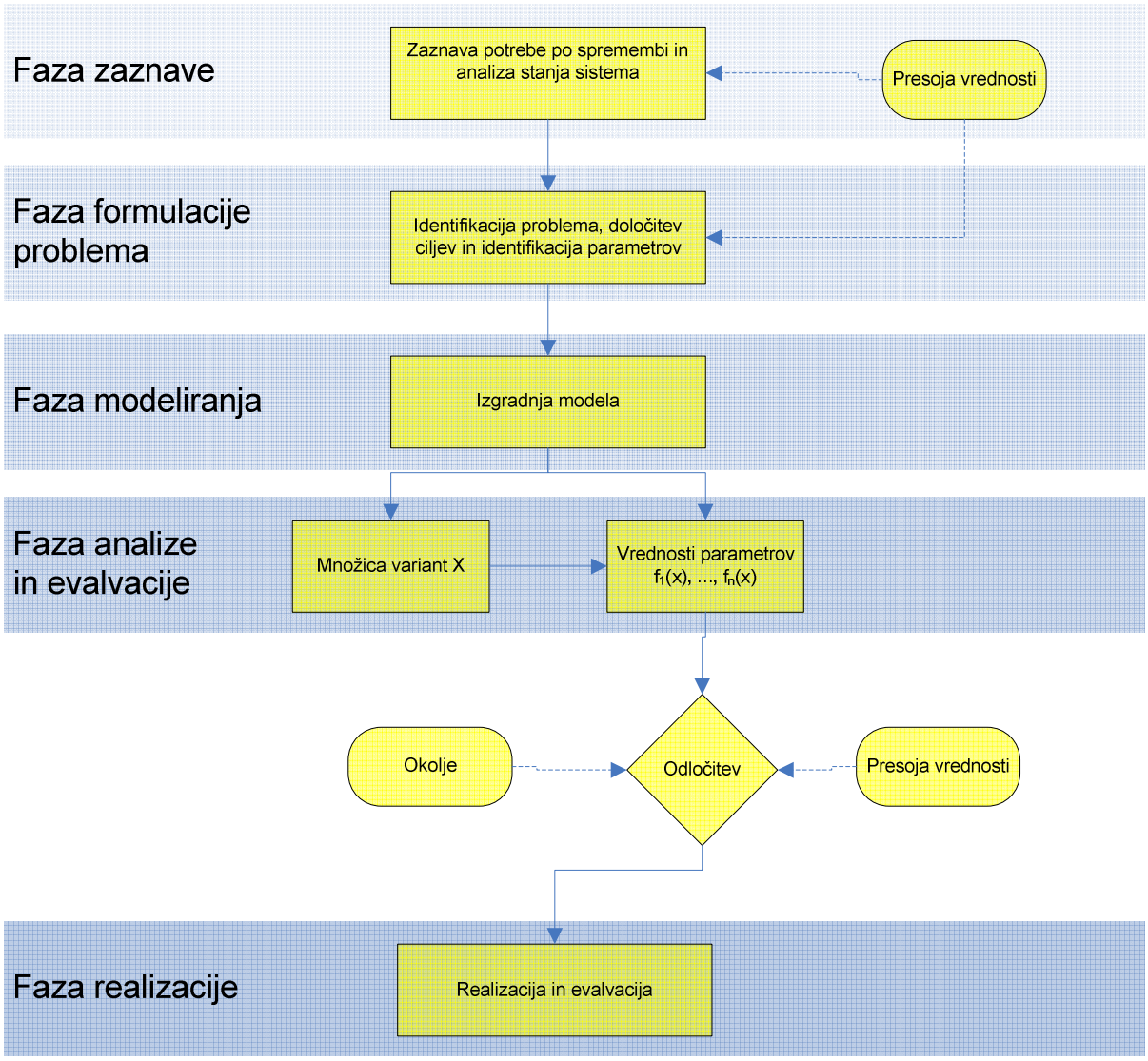
V praksi večkrat srečujemo neposredni pristop. Funkcijo koristnosti določi odločevalec po lastni presoji na osnovi izkušenj in prepričanj. Možno pa je tudi prepletanje neposrednega in aksiomskega pristopa. Gre za neposredno identifikacijo odločitvenega znanja, ki ga, če je le izvedljivo, tudi aksiomatsko utemeljimo.

4.6 Večparametrsko odločanje

Redko nastopajo primeri, kjer ocenjujemo variante le po eni lastnosti oziroma parametru. Vzemimo primer nakupa avtomobila. Lahko bi se odločili le na osnovi cene kot edinega parametra. Preferenčna relacija bi bila »je cenejši« in funkcija koristnosti obratna s ceno. V tem primeru bi se odločili za najcenejši avto.

Vemo pa, da cena še zdaleč ni edini parameter, kriterij oziroma odločitvena spremenljivka, ki vpliva na odločitev. Zanimajo nas na primer še varčnost, varnost pa tudi še kaj drugega. V takem primeru, ko odločamo na osnovi različnih pogledov na variante, govorimo o *večparametrskem odločanju*. Skoraj vsi realni odločitveni problemi so večparametrski. Srečamo odločitvene probleme z nekaj parametri pa vse do stotin parametrov pri kompleksnih odločitvenih situacijah.

Odločitveni postopek se začne z zaznavo potrebe po spremembi sistema in z identifikacijo stanja. Temu sledi formulacija problema, kjer se točno določijo cilji in identificirajo parametri. Ko so sistem, njegovo okolje in cilji dobro definirani, se prične izgradnja modela. Pri tem se lahko pojavijo različne oblike modelov, od preprostih miselnih in grafičnih do kompleksnih fizičnih in matematičnih. Temu sledi stopnja analize in evalvacije, kjer se vsaka varianta najprej oceni na osnovi prej določenih parametrov, nato pa se na osnovi teh ocen variante primerjajo med seboj. Varianta, ki ima glede na odločitveno pravilo najvišjo oceno, se nato realizira. Če smo z rezultatom zadovoljni, se postopek konča, drugače se vrnemo na stopnjo formulacije problema. Postopek je prikazan na sliki 4-1. Faze odločitvenega postopka pa so natančneje opisane v poglavju 4.9.



SLIKA 4-1: PROCES VEČPARAMETRSKEGA ODLOČANJA

V splošnem torej nastopa pri večparametrskem odločanju, kot pri navadnem odločanju množica variant *A* in preferenčna relacija *P*.

Pri večparametrskem odločanju imamo množico parametrov *X*: $x_1, x_2, x_3, \dots, x_n$

$$x_i : A \Rightarrow D_i \qquad (10)$$

kjer so D_i zaloge vrednosti posameznih parametrov.

Vsako varianto *a* iz *A* opišemo z naborom (vektorjem) vrednosti parametrov:

$$a = x_1(a), x_2(a), x_3(a), \dots, x_n(a) \qquad (11)$$

Preferenčna relacija *P*, ki uredi množico *A* po zaželenosti oziroma koristnosti, sedaj deluje med temi vektorji.

Funkcijo koristnosti: A ⇒ D nadomestimo s funkcijo

$$v_x: D_1 \times D_2 \times D_3 \times \ldots \times D_n \Rightarrow D \qquad (12)$$

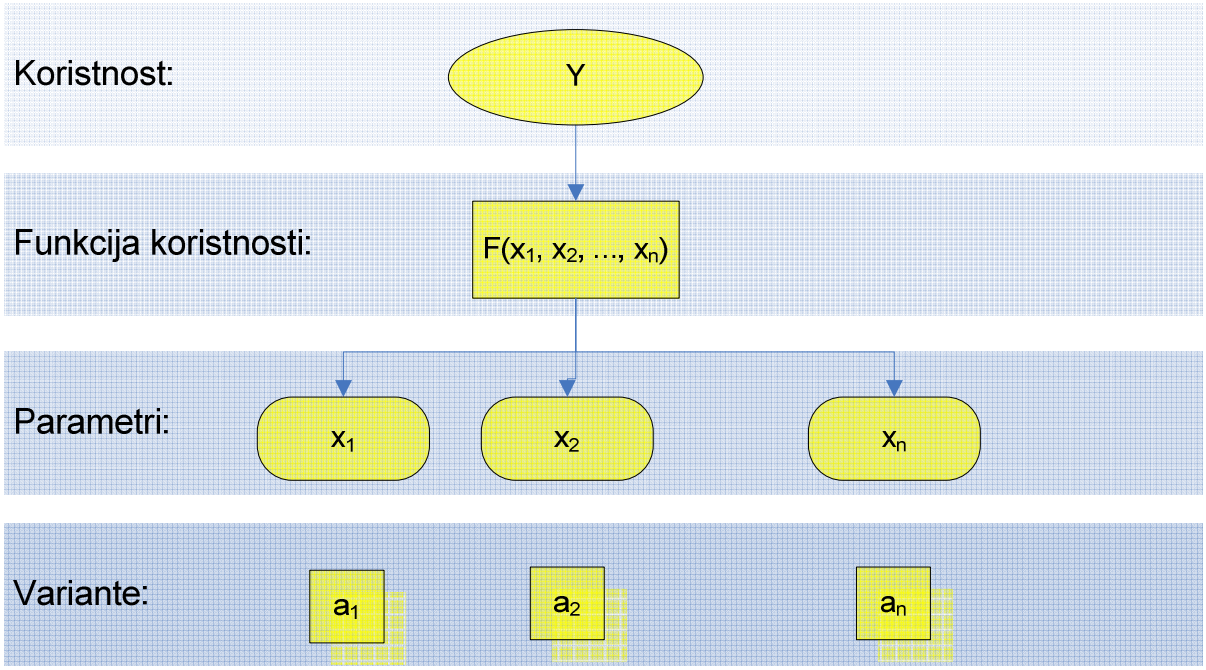
ki je definirana nad domeno, predstavljeno s kartezičnim produktom domen posameznih parametrov, in je *D* zaloga vrednosti funkcije koristnosti v_x.

Pri tem predpostavljamo, da je

$$v(a) = v_x(x_1(a), x_2(a), x_3(a), \ldots, x_n(a)) \qquad (13)$$

Funkcija koristnosti predstavlja "združeno" meritev koristnosti po vseh parametrih. Je kriterijska funkcija, s katero določamo koristnost variant na osnovi posameznih parametrov oziroma kriterijev in njih povezave.

Slika 4-2 prikazuje večparametrski odločitveni model, ki je v splošnem sestavljen iz zgoraj opisanih komponent, na osnovi katerega poteka vrednotenje variant pri večparametrskem odločanju.



SLIKA 4-2: VEČPARAMETRSKI ODLOČITVENI MODEL

Pri večparametrskem odločanju predpostavljamo opisljivost variant in obstoj ustrezne funkcije koristnosti.

4.7 Opisljivost variant

Tu gre v prvi vrsti za izbiro množice parametrov, s katerimi lahko opišemo vse karakteristike variant, ki vplivajo na odločanje. Različni avtorji navajajo različne vrste lastnosti in pogojev, ki jim mora ustrezati množica opisanih parametrov. Najpogosteje se navajajo lastnosti: polnost, operativnost, razstavljivost, neredundandnost in minimalnost [8].

Polnost je izpolnjena takrat, ko so vsi za odločanje vplivni dejavniki predstavljeni s parametri. V praksi težko vzpostavimo mehanizme, ki nam zagotavljajo potrebne in zadostne pogoje za tako polnost. V principu dodajanje novih parametrov že polnemu naboru ne bi smelo vplivati na vrednotenje. Nek nov parameter pa ne vpliva na vrednotenje le, če se da izraziti z že obstoječimi ali pa ker je v dani odločitveni situaciji pomemben.

Operativnost razumemo kot neposredno uporabnost v procesu odločanja. Najprej moramo za vsako varianto iz *A* dobiti vrednosti parametrov iz *X*. Vrednost vsakega parametra je lahko na primer:

- ena sama,
- verjetnostna porazdelitev ali
- pripadnostna porazdelitev v smislu mehkih množic.

Znotraj konceptov verjetnostnega oziroma mehkega opisa lahko do neke mere rešujemo tudi problem delne opisljivosti variant. Če ne poznamo vrednosti nekega parametra dane variante, v skrajnem primeru lahko predvidimo, da vse vrednosti parametra nastopajo z enako gotovostjo. V praksi pa se izkaže, da že ob površnem premisleku o posameznih vrednostih lahko povemo, katere so bolj gotove od drugih. Širina razmišljanja, ki nam jo ponujajo mehke množice, omogoči, da zberemo razpoložljive podatke in v grobem ločimo seme od plevela. Ob tem nam običajno postane tudi jasno, kaj bi morali še narediti, da bi mehkost ocene zmanjšali. Seveda nastopi vprašanje časovnih, materialnih in finančnih možnosti, če to zmoremo oziroma če se izplača.

Poleg opisljivosti posameznih konkretnih variant pa zahteva po operativnosti v splošnem predvideva tudi nadaljnjo »izračunljivost« v postopku vrednotenja.

Razstavljivost parametrov je lastnost, ki omogoča dekompozicijo odločitvenega problema v smislu izražanja nekega na primer težko merljivega parametra z dvema ali več laže merljivimi.

Neredundantnost predvideva, da je vsak odločitveni dejavnik zajet le z enim parametrom. Gre za medsebojno neodvisnost parametrov, ki je v praksi redka in težko preverljiva. Prav na tej lastnosti pa sloni veliko predpostavk o funkciji koristnosti. V okviru naše odločitvene metodologije omejena lastnost ni potreben pogoj pri izbiri parametrov.

Minimalnost predvideva, da ne obstaja drug opisno poln nabor parametrov kakega odločitvenega problema, ki ima manjše število parametrov. V našem primeru tudi to ni potreben pogoj za izbiro parametrov.

4.8 Strukturiranje parametrov

Če imamo parametrov veliko, na primer več deset ali več sto, jih ne moremo obravnavati naenkrat. Razlog je v nas samih, ker zmoremo v glavi držati le nekaj stvari hkrati in jih tudi obdelovati. V osnovi je princip strukturiranja osnovni princip reševanja problema. Vprašanje je le, kako strukturirati parametre.

Tudi v metodah, ki temeljijo na delnih funkcijah koristnosti, ne moremo shajati brez strukturiranja parametrov. Parametre razdelimo v skupine po nekaj, na primer največ do deset. Tako tudi laže določimo združeno funkcijo koristnosti, kjer moramo, na primer pri *aditivnem združevanju*, podati uteži kot odstotke pomembnosti posameznega parametra. Poskusimo razdeliti 100 odstotkov na deset parametrov in s tem podati njihovo relativno pomembnost. To napravimo mnogo laže pri manjšem številu parametrov, na primer pet.

Posamezne skupine parametrov ponovno združujemo na naslednjem nivoju običajno v skladu z drevesno strukturo.

Pri neposrednem združevanju je posebnost morda v tem, da združujemo parametre v sorazmerno majhne skupine od dva do štiri. Od strukture parametrov pričakujemo predvsem pomoč pri vsebinskem razmišljanju o parametrih, njih povezavi in vplivu na koristnost, to je končno oceno. Zato jih združujemo praviloma na osnovi vsebinske povezanosti.

Osnovna ideja je združevanje parametrov v nove, tako imenovane konceptualne parametre. Ti v splošnem niso neposredno merljivi na variantah, ampak so vsebinsko opredeljeni kot nov koncept, ki izhaja iz parametrov, ki jih le-ta združuje. Tako na primer koncept cene avtomobila opredeljujeta parametra nabavne cene in cene vzdrževanja. Ceno vzdrževanja pa lahko nadalje opredeljujejo cena rezervnih delov, cena servisne ure in poraba goriva.

Tako strukturiranje razbije problem na podprobleme v skladu z vsebinskimi celotami, ki jih lahko obvladamo in so med seboj smiselno povezane. To nas spominja na tehniko miselnih vzorcev. S tem pridemo ne le do manjših obvladljivih celot, ampak so te celote take, ki se skladajo z običajnim razmišljanjem o problemu.

Ni pa nujno, da strukturiranje poteka vedno od spodaj navzgor, to je, da združujemo posamezne parametre iz obsežnih množic parametrov. V principu lahko strukturiramo tudi od zgoraj navzdol. Končni združeni parameter, to je celotno oceno koristnosti sistema, skušamo razgraditi na vsebinsko pomembne sklope vse do merljivih parametrov. V praksi se ponavadi odločimo za kombinirano pot. Obstoječe parametre združujemo, če pa kateri med njimi neposredno ni merljiv ali ga želimo še nadalje razgraditi, napravimo to s skupino podparametrov.

4.9 Faze odločitvenega postopka

Faze odločitvenega postopka bomo skušali pogledati kot faze kibernetičnega postopka, v katerem želimo ustrezno upravljati proces odločanja. Ustreznost se nanaša na izbran kriterij ali pa kar na kriterijsko funkcijo, s katero ocenjujemo in izbiramo posamezne akcije pri izvajanju postopka vrednotenja in tega končni cilj je izbira najboljše variante.

Vemo, da je odločanje samo po sebi del vsakega kibernetičnega procesa, to je procesa upravljanja v živih in neživih sistemih. Pri tem mislimo na tehnične sisteme, na družbeno-ekonomske sisteme pa tudi na kombinirane organizacijske sisteme. V splošnem kibernetičnem procesu predstavljajo razpoložljiva upravljanja variante, ki jih ocenjujemo. Preferenčno relacijo med njimi skušamo predstaviti s kriterijsko funkcijo, ki v danem trenutku oceni variante in na osnovi te ocene izberemo najustreznejše upravljanje.

Postopek odločanja je splet posameznih faz, ki jih običajno razdelimo takole [4]:

1. **Identifikacija problema**

Ta faza je rezultat spoznanja, da je nastopil odločitveni problem, ki je dovolj težak, da ga je smiselno reševati na sistematičen in organiziran način. V tej fazi poskušamo definirati problem ter opredeliti cilje in zahteve.

Odločanje je ponavadi skupinsko delo. Četudi bi se odločali sami ali bi na koncu odločil nekdo tretji, je odločitvena skupina dobrodošla že v smislu "več ljudi več ve".

Naša želja je, da odločamo na osnovi čimbolj popolne informacijske slike, kajti na ta način lahko zmanjšamo možnost, da bi spregledali kaj pomembnega in se zato napak odločali. Zato je določitev odločitvene skupine, ki bo sodelovala v procesu odločanja, bistvenega pomena.

Jedro odločitvene skupine sestavljajo odločevalci (tako imenovani »lastniki problema«), to so tisti, ki se morajo v končni fazi odločiti in so odgovorni za odločitev. Pri zahtevnejših problemih je priporočljivo v delo skupine vključiti tudi:

- eksperte, ki imajo poglobljeno znanje o dani problematiki in lahko svetujejo pri oblikovanju odločitvenega modela, njihova vloga pa je neprecenljiva tudi pri argumentaciji odločitev in pri realizaciji izbrane variante;
- odločitvenega analitika-metodologa, ki kot moderator vpliva na učinkovitost in usklajenost dela skupine ter skrbi za ustrezno metodološko in računalniško podporo odločanja; po možnosti naj bi ta analitik ne prišel iz okolja, ki je neposredno prizadeto z odločitvijo;
- druge predstavnike tistih področij, na katere vpliva odločitev.

Variante ali alternative so možnosti, med katerimi izbiramo najustreznejšo ali najustreznejše. Do različnih variant včasih pridemo zelo preprosto, kot na primer pri prijavah na objavljeni razpis. Problem pa nastopi, ko moramo določiti variante sami. Pri tem je najpomembnejše, da nobena smiselna varianta ne ostane neopažena.

Na tej stopnji torej zberemo vse smiselne variante oziroma alternative. Variante običajno le evidentiramo in jih v odločitveni skupini še ne analiziramo, ker bi to lahko vplivalo na odločitvene parametre oziroma odločitveno znanje ter s tem na pristranskost odločitev.

2. **Identifikacija kriterijev (odločitvenih parametrov)**

V tej fazi določimo kriterije, na osnovi katerih bomo ocenjevali variante. in zasnujemo strukturo odločitvenega modela. Posebej je pomembno, da pri tem ne spregledamo kriterijev, ki bistveno vplivajo na odločitve (načelo popolnosti). Pri oblikovanju modela poskušamo izpolniti tudi nekatere druge zahteve, kot so strukturiranost, polnost, operativnost, neredundandnost, minimalnost kriterijev. Postopek identifikacije kriterijev je do neke mere odvisen od uporabljene metodologije. V našem primeru bo potekal po naslednjih korakih:

- Spisek kriterijev: sami ali med pogovorom v skupini bomo oblikovali nestrukturiran seznam kriterijev, ki jih bomo upoštevali pri odločanju.
- Strukturiranje kriterijev: kriterije bomo hierarhično uredili, upoštevajoč medsebojne odvisnosti in vsebinske povezave. Nepomembne kriterije in tiste, ki bodo izraženi z ostalimi kriteriji, bomo zavrgli in po potrebi oblikovali nove. Rezultat bo drevo kriterijev.
- Merske lestvice: vsem kriterijem v drevesu bomo določili merske lestvice, to je zalogo vrednosti, ki jih lahko zavzamejo pri vrednotenju, ter morebitne druge lastnosti (na primer urejenost).

3. **Zajemanje odločitvenega znanja – definicija funkcij koristnosti**

V tej fazi gre za tako imenovano določitev preferenc, ki povedo, katere lastnosti (parametri) oziroma njihove vrednosti so bolj zaželene od drugih. Nato je potrebno določiti, kako posamezne vrednosti parametrov vplivajo na končno vrednost koristnosti - oceno. Srečamo se z združevanjem (agregacijo) delnih ocen po posameznih parametrih v končno oceno. Parametre združujemo na osnovi vsebinskih medsebojnih povezav. Povezovanje delnih ocen v skupno oceno izrazimo z ustrezno aritmetično oziroma logično funkcijo.

Oblika funkcije in način zajemanja sta močno odvisna od uporabljene metode. Najpogosteje se uporabljajo preproste funkcije, kot so utežna vsota in razna povprečja. Srečamo pa tudi zahtevnejše funkcije, ki imajo večjo izrazno moč, vendar so nekoliko zahtevnejše za praktično uporabo. To so funkcije zvezne logike, funkcije na osnovi Bayesovega pravila ali mehkih množic, odločitvena pravila. Prav tako so pestre računalniško podprte metode za podporo odločevalcev v tej fazi in segajo od neposrednega analitičnega izražanja funkcij do možnosti izbiranja oziroma parametrizacije vnaprej pripravljenih funkcij, definiranja funkcij po točkah, zajemanja v grafični obliki in raznih dialogov, ki jih vodi računalniški program.

4. **Opis variant**

Vsako varianto opišemo z vrednostmi osnovnih kriterijev, to je tistih, ki ležijo na listih drevesa. Do tega opisa nas vodi bolj ali manj zahtevno proučevanje variant in zbiranje

podatkov o njih. Pri tem se pogosto srečamo s pomanjkljivimi ali nezanesljivimi podatki. Nekatere metode v tem primeru odpovedo, druge pa omogočajo, da takšne podatke opišemo v obliki intervalov ali verjetnostnih porazdelitev.

5. **Vrednotenje in analiza variant**

Vrednotenje variant je postopek določanja končne ocene variant na osnovi njihovega opisa po osnovnih kriterijih. Vrednotenje poteka »od spodaj navzgor« v skladu s strukturo kriterijev in funkcijami koristnosti. Varianta, ki dobi najvišjo oceno, je praviloma najboljša.

Besedo praviloma je potrebno na tem mestu posebej poudariti. Na končno oceno vpliva mnogo dejavnikov in pri vsakem od njih lahko pride do napake. Poleg tega sama končna ocena navadno ne zadostuje za celovito sliko o posamezni varianti. Zato moramo variante analizirati in poskusiti odgovoriti na naslednja vprašanja:

- Kako je bila izračunana končna ocena – na osnovi katerih vrednosti kriterijev in katerih funkcij? So vrednosti kriterijev in uporabljene funkcije koristnosti ustrezne?
- Zakaj je končna ocena takšna, kot je? Je v skladu s pričakovanji ali odstopa in zakaj? Kateri kriteriji so najbolj prispevali k takšni oceni?
- Katere so bistvene prednosti in pomanjkljivosti posamezne variante?
- Kakšna je občutljivost odločitve: kako spremembe vrednosti kriterijev vplivajo na končno oceno? Ali je mogoče variante izboljšati in kako? Katere spremembe povzročijo bistveno poslabšanje ocen variant?
- V čem se variante bistveno razlikujejo med seboj?

Šele z odgovori na ta vprašanja pridemo do celovite slike o variantah in s tem do kvalitetnejše, bolj utemeljene in preverjene odločitve. Računalniška podporna orodja so pri tem praktično nepogrešljiva, saj imajo že vgrajene pripomočke, ki tovrstne analize bistveno olajšajo.

To je stopnja, ki na prvi pogled ne sodi v proces odločanja. Vendar je od te stopnje in njene povezave s predhodnimi v dokajšnji meri odvisna kvaliteta odločitve. Gre za stopnjo realizacije odločitve, ki jo moramo spremljati tudi na osnovi kriterijev in odločitvenega znanja. Nenehno nas spremljajo spremembe. Če so le-te vplivne za našo odločitev, lahko ob ponovitvi nekaterih navedenih stopenj bistveno pripomoremo k zmanjšanju možnosti, da bomo pri realizaciji variante krenili po napačni poti.

Faze se lahko med seboj prepletajo, dopolnjujejo oziroma ciklično ponavljajo. V postopku odločanja pa nanje ne bi smeli pozabiti, saj gre za proces sistematičnega zbiranja in urejanja znanja - učni proces, ki naj bi zagotovil dovolj znanja za primerno odločitev, zmanjšal možnost, da bi kaj spregledali, organiziral in pocenil proces odločanja in seveda dvignil kakovost odločitve tako, da bi bila razumljiva in utemeljena.

4.10 AHP metoda

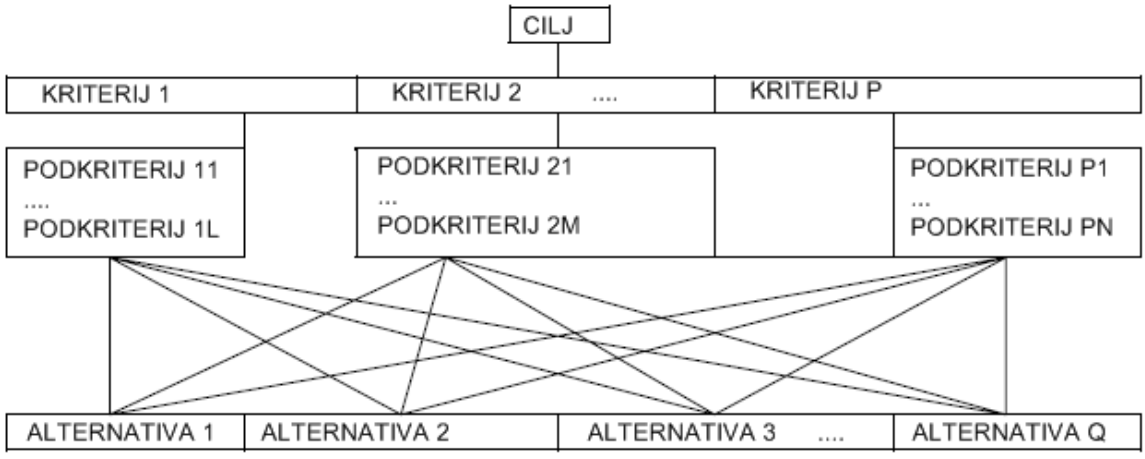
Metoda AHP (Analytic Hierarchiacal Process) je ena najbolj znanih in najbolj popularnih metod večparametrskega odločanja [4]. Posebnost metode AHP je, da uteži določimo posredno tako, da paroma primerjamo parametre, vsakega z vsakim. Podobno je pri zajemanju osnovnih koristnosti alternativ, ki jih določimo z medsebojno primerjavo alternativ. Ker osnovne koristnosti alternativ dobimo s primerjanjem, je metoda uporabna za vse vrste osnovnih parametrov (numerične in simbolične).

Osnovna struktura modela je pri AHP hierarhija parametrov, katerih vrednosti so izražene s preferenčno mersko lestvico na intervalu 0-1. Pri modelu AHP običajno narišemo tudi vse alternative in vsako od njih povežemo z vsemi osnovnimi parametri modela. To naredimo zato, ker metoda AHP zahteva, da v vseh vozliščih, pri katerih se stikajo povezave, ki vodijo iz podrednih vozlišč, definiramo matriko povezav. Tam, kjer se stikajo povezave, ki vodijo iz podrednih parametrov, med seboj primerjamo te parametre. Tam, kjer se stikajo povezave, ki vodijo iz alternativ, pa primerjamo alternative. Postopek primerjave in izračunavanja rezultatov je v obeh primerih enak.

Matriko primerjav izpolnimo tako, da primerjamo parametre, navedene v vrstici matrike, s tistimi v stolpcih. Iz navedenega sledi, da morajo biti vse diagonalne vrednosti v matriki enake 1, ker primerjamo parameter sam s sabo. Vrednosti, ki ležijo na eni in drugi strani diagonale, pa morajo biti med seboj obratne.

Metoda AHP je sosledje več korakov [3]:

Korak 1:
Problem razdelimo na hierarhijo ciljev, kriterijev, podkriterijev (parametrov) in alternativ. Ta korak je najbolj ustvarjalen in pomemben korak odločitvenega postopka. Strukturiranje odločitvenega postopka kot hierarhijo je bistveni del procesa AHP metode. Hierarhija predstavlja razmerja med elementi med višjim in nižjim nivojem. Značilna predstavitev te hierarhije je drevesna struktura. Na najvišjem nivoju hierarhije je cilj problema, ki ga preučujemo in analiziramo. Listi drevesa so alternative, ki jih primerjamo. Med tema dvema nivojema so različni kriteriji in podkriteriji. Važno je to, da ko se primerjajo elementi na nekem nivoju, se primerjajo glede na vrednosti spodnjega nivoja. Slika 4-3 prikazuje splošen primer hierarhične strukture.



SLIKA 4-3: HIERARHIČNA STRUKTURA KRITERIJEV

Odločevalec ocenjuje oz. primerja pomembnost parametrov samo na podproblemu in tega lahko natančneje oceni, kar je velika prednost AHP metode.

Korak 2:
Kriteriji so zbrani s pomočjo strokovnjakov na področju in odločevalcev. Kriteriji se ocenjujejo paroma. Ocenjuje se pomembnost parametra glede na drug parameter. Ta razmerja lahko pretvorimo v številčne vrednosti:

1 - parametra sta enako pomembna;
3 - zmerno večja pomembnost prvega parametra v primerjavi z drugim;
5 - velika, znatna prednost prvega parametra;
7 - zelo velika prednost prvega parametra;
9 - skrajna, izjemna prednost prvega parametra.

Vmesne vrednosti 2, 4, 6 in 8 pomenijo kompromisne vrednosti med ostalimi vrednostmi. Kadar je pomembnejši drugi parameter, vpišemo ustrezno obratno vrednost (ulomek 1/3, 1/5, 1/7, 1/9).

Korak 3:
Vsa medsebojna primerjanja parametrov oblikujejo kvadratno matriko primerjav, katere diagonala je 1. Parameter v *i*-ti vrstici je pomembnejši od parametra v *j*-tem stolpcu, če je vrednost elementa (*i*,*j*) večja od ena. Sicer je parameter v *j*-tem stolpcu pomembnejši od *i*-te vrstice. Element (*j*,*i*) je recipročna vrednosti elementa (*i*,*j*).

Korak 4:
Izračuna se največjo lastno vrednost matrike primerjav in njen pripadajoč lastni vektor. Elementi lastnega vektorja nam povedo relativno pomembnost posameznega parametra glede na vse druge parametre. Lastni vektor normaliziramo in dobimo uteži posameznega parametra glede na podkriterij, ki ga rešujemo.

Korak 5:
AHP metodologija uporablja indeks konsistentnosti za merjenje konsistentnosti za vsako matriko primerjav. Konsistentnost pomeni, da je pri izboru članov član B primernejši kot član C in član C primernejši kot D, potem je pričakovati, da bo član B veliko primernejši kot član D. *Indeks konsistentnosti* (*CI*) in *razmerje konsistentnosti* (*CR*) matrike *A* sta definirana kot:

$$CI = \frac{(\lambda_{max} - n)}{n-1} \qquad (14)$$

$$CR = \frac{CR}{ACI} 100\% \qquad (15)$$

λ_{max} največja lastna vrednost matrike *A*,
n dimenzija matrike *A*,
ACI povprečni indeks glede na naključno generirane uteži je odvisen od *n*-ja [5]:

n	3	4	5	6	7	8	9	10	>10
ACI	0,52	0,89	1,11	1,25	1,35	1,40	1,45	1,49	1,56

V primeru, da je razmerje konsistentnosti manjše od 10%, potem je konsistentnost matrike medsebojnih primerjav sprejemljiva. Sicer je potrebno popraviti matriko primerjav.

Korak 6:
Vrednost vsake alternative je pomnožena z utežjo vsakega podkriterija in potem združena v vsoto, da dobimo *lokalno vrednost* vsakega kriterija. Lokalne vrednosti so pomnožene z utežmi vsakega kriterija in potem dobimo *glavno vrednost* – skupno oceno alternative.

4.11 MAUT metoda

S kratico MAUT označujemo večjo in zelo pomembno skupino večparametrskih metod, osnovanih na teoriji večparametrske koristnosti. Metod tipa MAUT je več, vse pa delujejo po naslednjih načelih [4]:

1. Strukturo odločitvenega problema opisujejo z drevesom ali hierarhijo parametrov. V splošnem v taki strukturi nastopajo nadredni in podredni parametri. Vsak nadredni parameter se v splošnem deli na več podrednih parametrov, ki nastopajo na nižjem nivoju strukture modela. Delitev pomeni, da je nadredni parameter odvisen od podrednih parametrov; njegova vrednost se pri vrednotenju alternativ izračuna na osnovi podrednih parametrov. Parametre, ki nastopajo v strukturi, delimo na osnovne in izpeljane. Osnovni parametri so tisti, ki nimajo podrednih parametrov. V strukturi nastopajo kot končna vozlišča ali »listi«. Predstavljajo vhodne parametre modela vrednotenja. Preostali parametri so izpeljani. Vsi so nadredni in nastopajo kot notranja vozlišča strukture. Struktura modela je bodisi drevo bodisi hierarhija. Značilnost drevesa parametrov je v tem, da vsak podredni parameter vpliva na en sam nadredni parameter. Imamo čisto razdelitev parametrov z višjih na nižje nivoje. Hierarhija parametrov je nekoliko splošnejša; pri njej dopustimo, da kak podredni parameter vpliva na enega ali več parametrov. V matematičnem smislu je hierarhija definirana kot usmerjen aciklični graf.
2. Vsa notranja vozlišča strukture (izpeljani parametri) so zvezne spremenljivke, ki uporabljajo enotno preferenčno mersko lestvico. To je lestvica, predstavljena z intervalom [0, 1] (včasih tudi [0, 10] ali [0, 100]). Spodnja meja intervala ustreza izrazito slabi oziroma nezaželeni vrednosti parametra, zgornja pa idealni, najbolj zaželeni.
3. Vhodni parametri modela (osnovni parametri) so zvezne spremenljivke, ki jih izražamo in merimo v naravnih enotah: višino plače v denarnih enotah, površino zmeljišča v m^2, porabo goriva v litrih na kilometer.
4. Vrednotenje alternativ poteka z združevanjem vrednosti od spodaj navzgor, se pravi v smeri od osnovnih parametrov proti korenu strukture modela. Združevanje poteka v skladu s funkcijami koristnosti. Te so dveh vrst (prav zato za metode MAUT pogosto pravimo, da so dvostopenjske):
    - Koristnost vsakega posameznega parametra določimo z osnovnimi funkcijami koristnosti. Gre za preslikavo, ki »pravo« vrednost parametra (plačo, površino, porabo) preslika v stopnjo njegove zaželenosti (preferenco). Funkcijo imenujemo osnovna zato, ker se nanaša na osnovne parametre modela. Te preslikave določi odločevalec in so subjektivne, saj izhajajo iz njegovih ciljev in izražajo njegove želje, zahteve in pričakovanja.
    - Združevanje preferenc od spodaj navzgor opravimo s funkcijami združevanja. Pri tem najpogosteje uporabimo kar uteženo vsoto. Točke, ki jih dobi vsaka alternativa pri vsakem parametru, pomnožimo z utežjo tistega parametra. Tako dobljene zmnožke seštejemo.

Hierarhična struktura pri MAUT metodi pomeni, da se poleg končne ocene pri vrednotenju alternativ izračunajo tudi vrednosti vseh izpeljanih parametrov (notranjih vozlišč modela). Izpeljani parametri so torej hkrati tudi izhodni parametri modela. To je zelo pomembno, saj jih lahko opazujemo in analiziramo ter s tem pridobimo nov, boljši vpogled v posamezne komponente odločitve oziroma odločitvene podprobleme.
Druga pomembna lastnost metod tipa MAUT je bolj realno obravnavnaje osnovnih parametrov. Te sedaj izražamo in merimo s pravimi enotami, kakršne srečamo v resničnem življenju. Zaželenost teh vrednosti za odločevalca pa izražamo z osnovnimi funkcijami koristnosti. Te so v splošnem nelinearne in jih je zato možno oblikovati dokaj poljubno in v skladu z željami odločevalca.

5 MODEL ZA SESTAVO VIRTUALNEGA TIMA

5.1 Identifikacija odločitvenega problema

Dandanes se programerske hiše pogosto soočajo z zaposlovanjem pravilnega kadra, ki ustreza hitrim spremembam v razvojnih tehnologijah in spreminjajočim se projektnim trendom. Znanje kadra v programerskih tehnologijah je hitro zastarano in nemogoče je, da en človek znova in znova obnavlja svoje strokovno znanje na svojem področju in ostaja v stiku z najnovejšo tehnologijo. Prav tako pa je potrebno ohraniti stara znanja, saj razviti produkti zahtevajo vzdrževanje po več let. Klasična rešitev je začasno zaposlovanje novih ljudi ter izobraževanje že zaposlenih, katerih znanje je že preveč zastarano. Težava je, da je težko dobiti dovolj novega kadra z zahtevanim novim znanjem. Izobraževanje ljudi, ki so izgubili stik z moderno tehnologijo, je drago in težavno. Ob koncu projekta pa se nam sprostijo zaposleni, ki nam niso več potrebni.
Alternativa tej rešitvi je sestava virtualnega tima [20]. Glavna prednost te rešitve je, da kader izberemo iz velikega kroga iskalcev dela, ki imajo točno zahtevana znanja in sposobnosti. Virtualni tim se sestavi za izdelavo točno določenega projekta in se po končanju razpusti. Virtualni tim je sestavljen iz posameznih članov. Člane se analizira in ovrednoti na portalu ponudb dela. Predmet odločanja oziroma vrednotenja v tem primeru so torej iskalci zaposlitve. Cilj odločitvenega procesa pa je razviti model za vrednotenje in analizo iskalcev zaposlitve ter ga uporabiti za sestavo virtualnih timov.

5.2 Identifikacija kriterijev

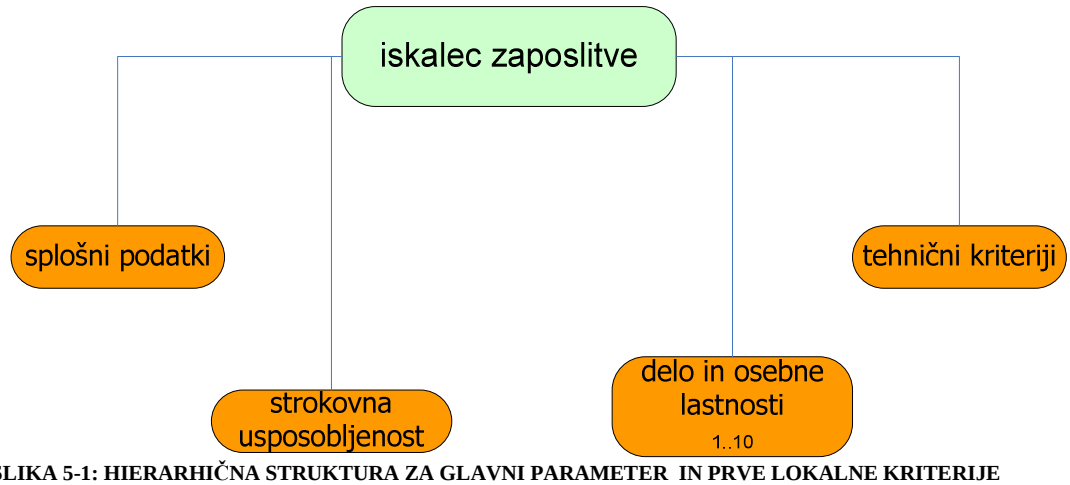
V tej fazi določimo kriterije, po katerih bomo ocenjevali variante, in zasnujemo strukturo odločitvenega modela. Spisek kriterijev je bil podan v poglavju 2.3.
Kriterije je potrebno hierarhično urediti, tako da nastane drevo (hierarhija) kriterijev. Najvišji izpeljani kriterij je glavni izhodni parameter modela in predstavlja glavno vrednost, skupno oceno alternative, v tem primeru je to končna ocena kandidata. Prvi podredni parametri, ki vplivajo na odločitev, pa so: splošni podatki, strokovna usposobljenost, delo, osebnostne lastnosti in tehnični kriteriji.

Določiti moramo merske lestvice vsem listom v drevesu. Večinoma ocenjujemo parametre z vrednostmi od 1 do 10. V primerih izključujočih vrednosti pa uporabimo vrednost 0 oz. 1.

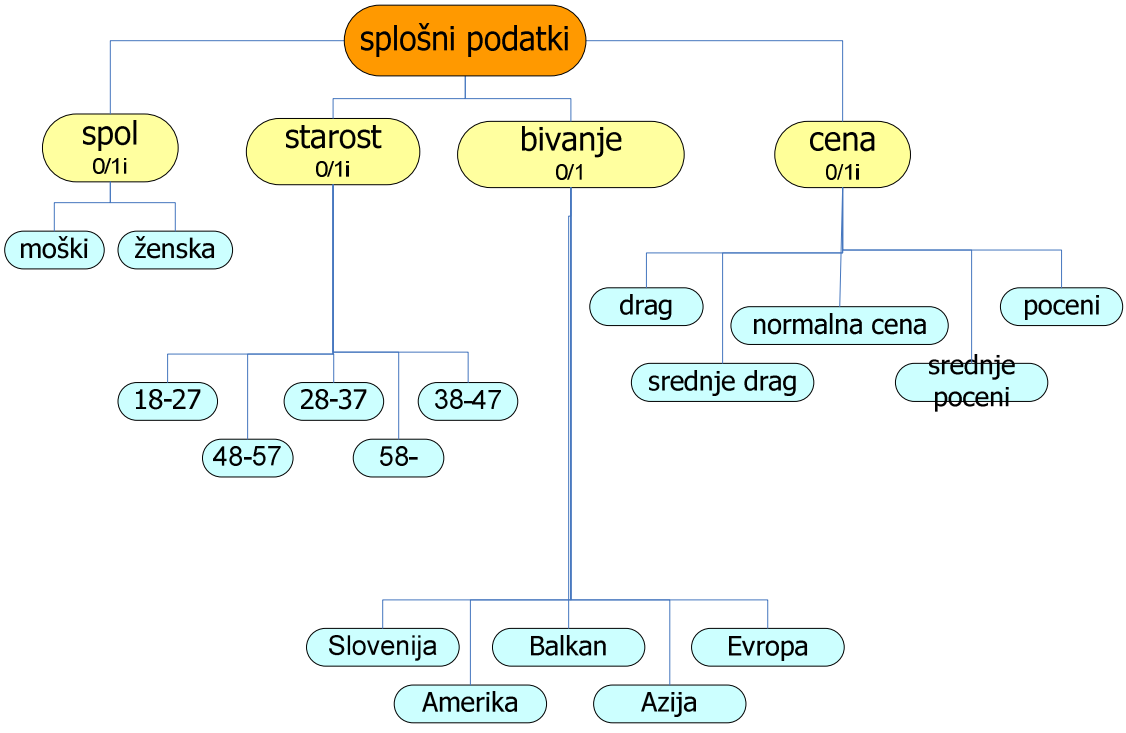
Tak primer je spol. V primeru, da je spol moški, ima parameter vrednost 1. Posledično mora imeti parameter za ženski spol vrednost 0.

Drevo kriterijev, ki nastane, je na slikah od 5-1 do 5-5. Na sliki so vrednosti parametrov označene z:

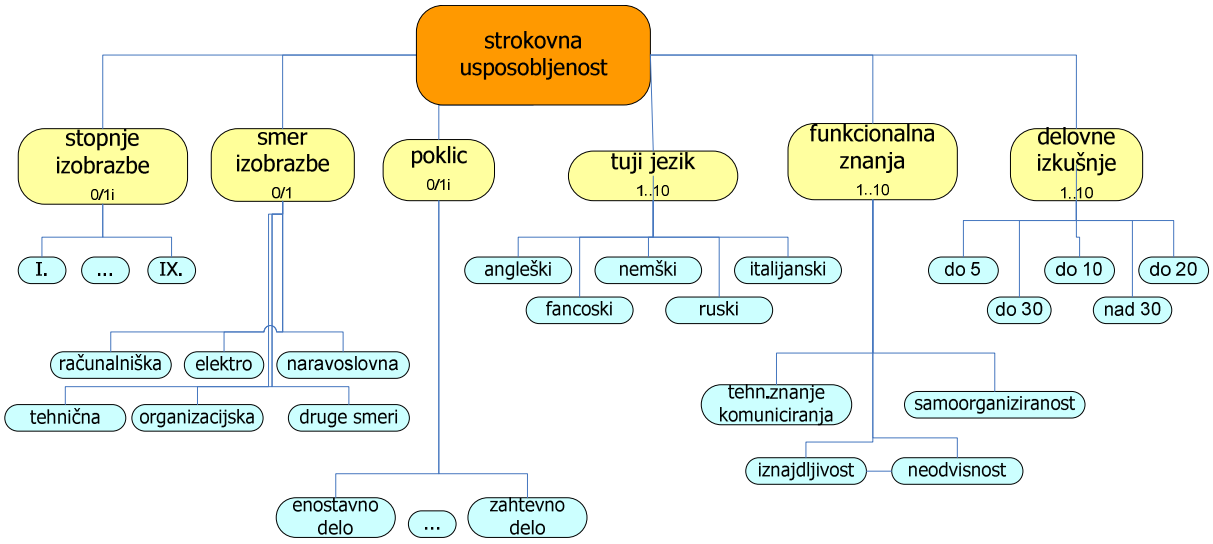
- 1..10 – vrednosti od 1 do 10;
- 0/1 – oseba ima karakteristiko tega parametra ali pa je nima;
- 0/1i - oseba ima karakteristiko tega parametra ali pa je nima; podredne vrednosti so si izključujoče.



SLIKA 5-1: HIERARHIČNA STRUKTURA ZA GLAVNI PARAMETER IN PRVE LOKALNE KRITERIJE

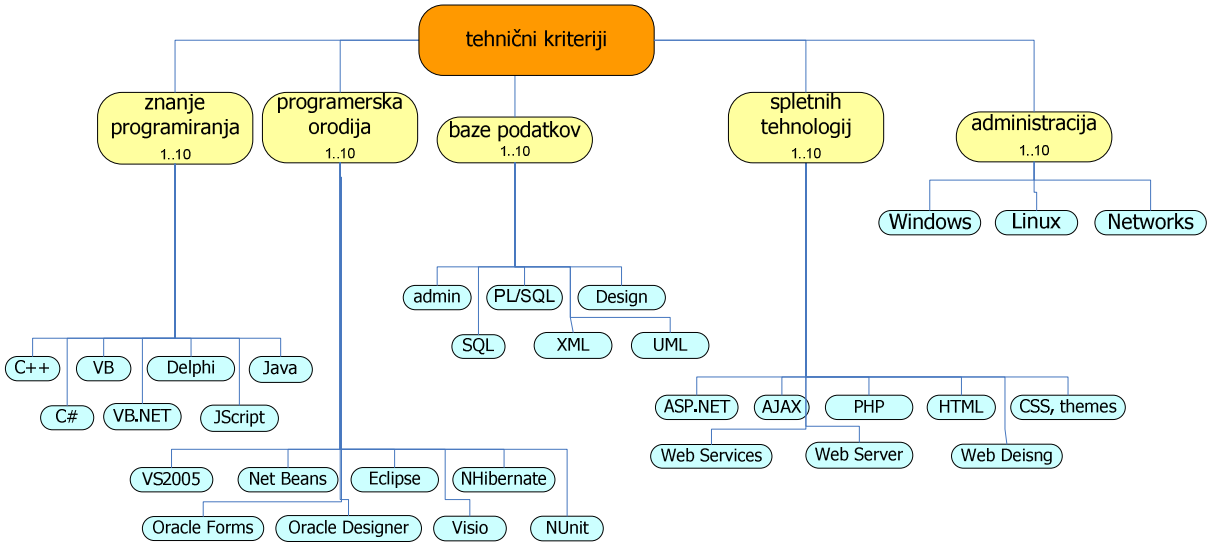


SLIKA 5-2: HIERARHIČNA STRUKTURA ZA PODKRITERIJ SPLOŠNI PODATKI



SLIKA 5-3: HIERARHIČNA STRUKTURA ZA PODKRITERIJ STROKOVNA USPOSOBLJENOST

SLIKA 5-4: HIERARHIČNA STRUKTURA ZA PODKRITERIJ DELO IN OSEBNE LASTNOSTI



SLIKA 5-5: HIERARHIČNA STRUKTURA ZA PODKRITERIJ TEHNIČNI KRITERIJI

5.3 Zajemanje odločitvenega znanja

V tej fazi določimo preference, ki povedo, katere lastnosti (parametri) oziroma njihove vrednosti so bolj zaželene od drugih. Pri tem uporabimo AHP metodo, ki zajame pomembnost določenih kriterijev problema.

Prav tako uporabimo AHP metodo pri določanju, kako posamezne vrednosti parametrov skozi hierarhijo vplivajo na končno vrednost koristnosti – oceno. Glede na hierarhijo parametrov, ki smo jo zgradili v prejšnjem podpoglavju, bomo za vsako vozlišče, kjer se stikajo povezave, ki vodijo iz podrednih vozlišč, definirali matriko povezav. V njej bomo ocenjevali pomembnost parametra glede na druge parametre. Matrike povezav se nahajajo v prilogi A.

Za vsako matriko povezav dobimo iz njenega lastnega vektorja (glede na maksimalno lastno vrednost) relativno pomembnost posameznega parametra glede na druge parametre.

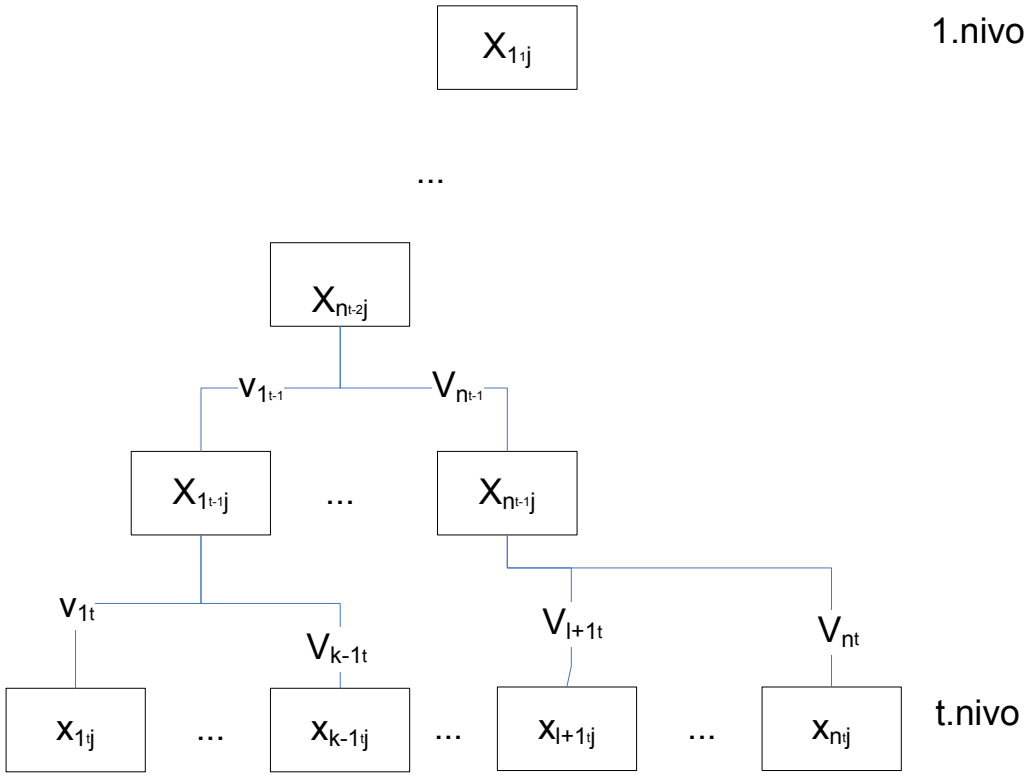
5.4 Vrednotenje in analiza variant

Končno oceno variant (alternativ) bomo podali na osnovi opisov, ki so jih dali kandidati skozi zaposlitveni portal. Vrednotenje poteka po MAUT metodi (poglavje 4.11) od spodaj navzgor v skladu s strukturo, podano v poglavju 5.2. Za funkcijo združevanja uporabimo kar združevanje (agregacijo) delnih ocen po posameznih podkriterijih v oceno kriterija. Za združevanje uporabimo funkcijo uteženih vsot, to je, da točke, ki jih dobi vsaka alternativa pri vsakem parametru, pomnožimo z utežjo tistega parametra. Tako dobljene zmnožke seštejemo.

Imamo *t* nivojev.

Relativno pomembnost parametra i_s za kandidata j na nivoju s označimo z v_{i_s}. Če so vrednosti parametra i_t za kandidata j na nivoju t enake $x_{i_t j}(a)$, potem velja: $x_{i_{t-1} j}(a) = v_{k_t} * x_{k_t j}(a) + \ldots + v_{l_t} * x_{l_t j}(a)$, kjer je $x_{i_{t-1} j}(a)$ lokalna vrednost nadkriterija i_{t-1} glede na podkriterije $i_t = k \ldots l$ $(k \geq 1 \text{ in } l \leq n)$. Rekurzivno tako izračunamo vrednosti vseh nadkriterijev na podlagi vrednosti in relativnih vrednosti podkriterijev:

$$x_{i_{s-1} j}(a) = v_{k_s} * x_{k_s j}(a) + \ldots + v_{l_s} * x_{l_s j}(a) \qquad (16)$$



SLIKA 5-6: OCENJEVANJE ALTERNATIV PO HIERARHIJI

Te vrednosti uporabimo za izračun glavne vrednosti (cilja) kandidata $x_{1_1 j}(a)$.

5.5 Problem linearnega programiranja

Z odločitvenim modelom se je postavil sistem za ocenjevanje in odločanje med različnimi kandidati. Za sestavo virtualnega tima moramo uporabiti optimizacijsko metodo, ki bo izbrala med vsemi kandidati tiste, katerih kombinacija karakteristik bo najbolje ocenjena. Problem sestave virtualnega tima je predstavljen kot primer celoštevilčnega linearnega programiranja. Model temelji na karakteristikah in na njihovih utežeh, ki jih pridobimo iz AHP metode.

Najprej identificiramo odločitvene spremenljivke. Glede na število podatkov

i število karakteristik (do *n*),
j število kandidatov (do *m*)

bomo potrebovali *i* * *j* neodvisnih spremenljivk *t*

t_{ij} kandidat *j* je izbran zaradi karakteristike *i* (izbran za nalogo *i*).

Nekatere karakteristike so take, da morajo biti ocenjene pri vseh kandidatih, nekatere pa želimo samo pri enem kandidatu. Vrednost *p* nam pove, koliko je takih karakteristik, ki jih ocenjujemo pri vseh kandidatih.

Preden identificiramo omejitve, da jih bomo lahko zapisali v obliki linearnih enačb oz. neenačb, identificiramo še parametre *w*, ki jih dobimo iz AHP metode:

$w_{i_t j}$ ocena kandidata *j* glede na potrebo poizvedovalca za karakteristiko *i* na najnižjem nivoju. Ocena se izračuna iz uteži, ki se dobijo iz AHP metode. Zmnoži se koeficiente na vseh nivojih, tako da je

$$w_{i_t j} = \left(\sum_{s=1}^{t} v_{i_s j}\right) * x_{i_t j}(a) \qquad (17)$$

Omejitve formuliramo po točkah:

1. Vsaka oseba j ima maksimalno m_j izbranih karakteristik:

$$\forall\, j \;:\; \sum_{i=1}^{n} t_{ij} \;\leq\; m_j + p \qquad (18)$$

2. Kriterij *i* ima lahko D_i različno število oseb:

   - (kriterij *i* ima največ D_i ljudi)

$$\sum_{j=1}^{m} t_{ij} \;\leq D_i \qquad (19)$$

   - (kriterij *i* ima natanko D_i ljudi; če morajo biti po kriteriju *i* ocenjeni vsi kandidati, je $D_i = m$)

$$\sum_{j=1}^{m} t_{ij} \;=\; D_i \qquad (20)$$

   - (kriterij *i* ima D_i ljudi ali več)

$$\sum_{j=1}^{m} t_{ij} \geq\; D_i \qquad (21)$$

3. V skupini želimo imeti največ T ljudi. Za ta pogoj definiramo dodatne neodvisne spremenljivke u_i, s čimer dosežemo pogoj, da lahko preštejemo kandidate, ki imajo lahko eno ali več karakteristik. Neodvisna spremenljivka u_j nam pove, ali je kandidat j izbran v tim ($u_j = 1$) ali ne ($u_j = 0$).

$$\forall j : \sum_{i=1}^{n} t_{ij} - u_j * n \leq 0 \quad (22)$$

$$\forall j : u_j - \sum_{i=1}^{n} t_{ij} \leq 0 \quad (23)$$

$$\forall j : u_j \leq 1 \quad (24)$$

$$\sum_{j=1}^{m} u_j \leq T \quad (25)$$

$$\forall i, \forall j : t_{ij} \leq 1 \quad (26)$$

Na koncu še identificiramo namensko funkcijo, ki jo bomo maksimizirali s spreminjanjem odločitvenih spremenljivk:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{m} w_{i_t j} * t_{ij} \quad (27)$$

5.6 Implementacija iskalnega algoritma

Iskalni algoritem teče po naslednjih točkah:

1. **Naložimo matrike povezav.**

   Matrike povezav, ki predstavljajo vse podkriterije in kriterije, predstavimo kot dvodimenzionalen double array. Za vsak kriterij in podkriterij dobimo svojo matriko povezav (poglavje 5.3 in priloga A).

2. **Izračunamo lastne vrednosti za matriko in najdemo največjo.**

   Za operacije z matrikami uporabimo Math.NET [14]. To je matematični odprtokodni sistem, implementiran v jeziku C# na Microsoftovi .NET platformi. Math.NET ima cilj postati samostojno ogrodje za numerično in simbolično računanje, ki ne potrebuje dodatnih zunanjih knjižnic.

   Za naše potrebe smo uporabili Iridium knjižnico v sklopu Math.NET ogrodja. Iridium je osnovna knjižnica, ki vsebuje splošno uporabne elemente za numerično računanje.

Ponuja nam ogrodje za linearno algebro, generatorje naključnosti in porazdelitve, posebne funkcije, hitro fouriejovo transformacijo (FFT) itd.

Knjižnica se bo uporabila za izračun lastnih vrednosti in lastnih vektorjev za potrebe AHP metode. Za to se bo uporabil razred EigenvalueDecomposition, ki že v konstruktorju izračuna vse lastne vrednosti in jih zapiše v listo:

Public ReadOnly Property EigenValues As Complex()

Za vsako matriko povezav uporabimo zgornji razred za izračun lastnih vrednosti in izmed vseh najdemo največjo lastno vrednost λ_{max}.

3. **Izračunamo razmerje konsistentnosti.**

Za vsako matriko povezav *A* in pripadajočo lastno vrednost λ_{max} izračunamo konsistentnost matrike s formulo, ki je opisana v 5. koraku AHP metode (poglavje 4.10).

Če konsistentnost matrike medsebojnih povezav ni sprejemljiva, se mora spremeniti ocena pomembnosti parametrov za ta kriterij oz. podkriterij. Algoritem se prekine.

4. **Izračunamo lastni vektor za največjo lastno vrednost.**

Za vsako matriko povezav *A* in pripadajočo lastno vrednost λ_{max} izračunamo pripadajoči lastni vektor v_{max}. Tu se zopet uporabi razred EigenvalueDecomposition, kjer so lastni vektorji izračunani v listi :

Public ReadOnly Property EigenVectors As Matrix

Vektor v_{max} še normaliziramo. S tem dobimo uteži za vsak parameter.

5. **Izvedemo eliminacijo kandidatov, ki so glede na sokandidate slabši.**

Uvedemo relacijo biti boljši ali enak med kandidati. Kandidat *p* je *boljši ali enak* od kandidata *r*, če za vse karakteristike $i = 1..n$ velja $w_{ip} \geq w_{ir}$.

V primeru, da izbiramo kandidate za tim velikosti *T*, potem lahko izvedemo operacijo: če za kandidata *r* obstaja *T* kandidatov, ki so boljši ali enaki, potem kandidata *r* izločimo iz kandidature za tim.

6. Sestavimo LP problem in ga rešimo.

Za reševanju problema sestave optimalnega tima je predpostavka o zveznosti odločitvenih spremenljivk nesprejemljiva. Za člana tima ne moremo vzeti 1,5 prvega člana in 2,3 drugega člana. Zato moramo postaviti zahtevo po celoštevilskih reštivah. Za reševanje takega problema se uporabi metodo celoštevilskega linearnega programiranja. Vse predpostavke pri metodah so enake kot pri navadnem linearnem programiranju, samo vrednosti odločitvenih spremenljivk bodo pri rešitvi imele celoštevilske vrednosti.

Za reševanje problema po metodi celoštevilskega linearnega programiranja se uporablja odprtokodni program LP – Solve [12], ki za reševanje uporablja Branch-and-bound algoritem. LP – Solve nima omejitve glede na velikost modela. Z večanjem modela pa se reševanje otežkoča ali pa celo rešitev ni mogoča.
LP – Solve lahko vrne več vrst izhodov:

NOMEMORY (-2)	Aplikaciji je zmanjkalo spomina, rešitev ni mogoča.
OPTIMAL (0)	Izračunana je bila ena optimalna rešitev LP problema.
SUBOPTIMAL (1)	Izračunana je rešitev LP problema, ki je v območju dopustnih rešitev. Rešitev pa ni nujno optimalna, Do tega pride v primeru: • ko preteče čas, ki je določen za maksimalno reševanje problema in je že najdena vsaj ena dopustna rešitev; • če je pri reševanju določeno, da naj se algoritem ustavi ob prvi najdeni dopustni rešitvi; • ob raznih klicih prekinitvene funkcije in je že najdena vsaj ena dopustna rešitev; • če zmanjka spomina in je že najdena vsaj ena dopustna rešitev.
INFEASIBLE (2)	LP nima nobene možne rešitve.
UNBOUNDED (3)	LP ima neomejeno množico rešitev.
DEGENERATE (4)	The model is degenerative.
NUMFAILURE (5)	Prišlo je do napake pri izračunu.
USERABORT (6)	Prišlo je do klica prekinitvene funkcije.
TIMEOUT (7)	Pretekel je čas, ki je določen za maksimalno reševanje problema.
PRESOLVED (9)	The model could be solved by presolve. This can only happen if presolve is active via set_presolve.
PROCFAIL (10)	V algoritmu Branch-and-bound je prišlo do napake.
PROCBREAK (11)	Algoritem Branch-and-bound se je ustavil zaradi prekinitvene zahteve.
FEASFOUND (12)	Najdena je bila rešitev po algoritmu Branch-and-bound.
NOFEASFOUND (13)	Ni bila najdena rešitev po algoritmu Branch-and-bound.

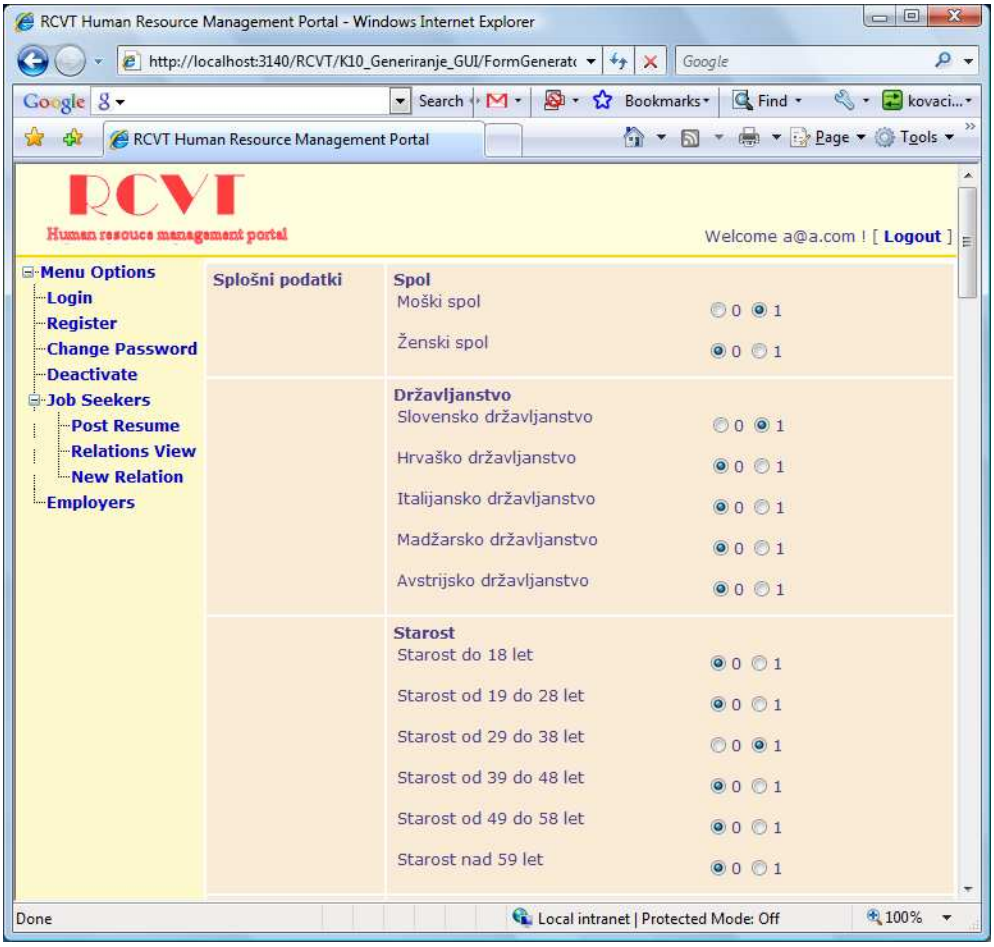
Kljub temu da ima linearno programiranje polinomsko časovno zahtevnost reševanja, pa tega celoštevilsko linearno programiranje nima. V veliko realnih primerih je celoštevilsko linearno programiranje NP poln problem.

6 PORTAL ZA KADROVANJE

Na podlagi analize in opisane tehnologije v tretjem poglavju smo implementirali spletni portal kadrovskega sistema.

Funkcionalnost portala se razlikuje glede na prijavljenega uporabnika. Če se v portal prijavi kot prosilec, ima funkcionalnost, ki se veže samo na vlogo prosilca:

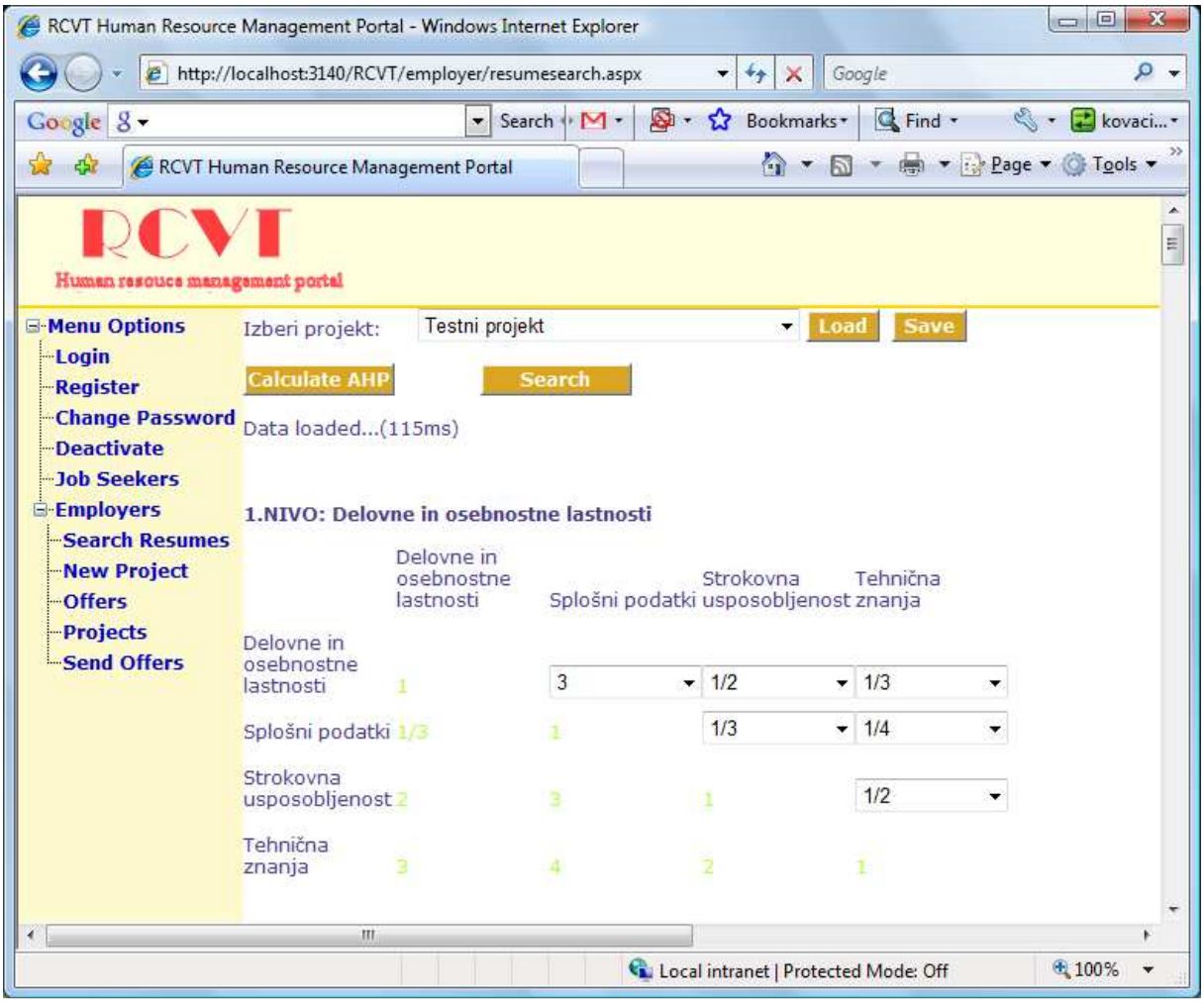
- Urejanje svojega življenjepisa (Slika 6-1);
- Pregled svojih znanstev;
- Dodajanje znanstev.



SLIKA 6-1: VNOSNA MASKA ZA ŽIVLJENJEPIS

Ob prijavi uporabnika, ki išče kader za sestavo tima za nek projekt, se v portalu pojavijo naslednje aktivnosti:

- Iskanje tima (Slika 6-2);
- Vnos novega projekta;
- Pregled zavrnjenih sprejetih ponudb;
- Pregled in urejanje projektov;
- Pošiljanje ponudb za delo posameznemu iskalcu zaposlitve.



SLIKA 6-2: VNOS KRITERIJEV ZA PROJEKT

Na strani za iskanje tima (Slika 6-2), lahko po vpisu ocen različnih variant izračunamo samo relativne pomembnosti posameznega parametra glede na druge parametre. Preverimo, ali so bile ocene pravilno vnešene in ali so relativne pomembnosti razdeljene tako, kot so bile okvirno zamišljene. Sicer lahko ocene popravimo in postopek ponavljamo, dokler ni dosežena željena iskalna pomembnost parametrov. Nato nad množico uporabnikov v podatkovni bazi izvedemo iskanje.

Za primer uporabe podamo potrebo dveh projektov. Prikažemo vnos potreb po kadru v portal in nato analiziramo rezultate iskanja.

Primer 1:

Iščemo tim dveh ljudi za programiranje v C#. Pomembno je, da imata malo delovnih izkušenj in predvsem veliko znanja v C#, prednost je tudi boljše znanje Visual Basica.NET.

Za ta primer delovne in osebnostne lastnosti ter strokovna usposobljenost niso potrebne, zato jih postavimo na najmanjšo prioriteto. Največjo pomembnost damo splošnim podatkom (ki vsebujejo delovne izkušnje), in sicer da je to 3-krat pomembnejše kot tehnično znanje.

1.NIVO: Delovne in osebnostne lastnosti

	Delovne in osebnostne lastnosti	Splošni podatki	Strokovna usposobljenost	Tehnična znanja
Delovne in osebnostne lastnosti	1	1/9	1	1/9
Splošni podatki	9	1	9	3
Strokovna usposobljenost	1	1/9	1	1/9
Tehnična znanja	9	1/3	9	1
	0,0693	0,8653	0,0696	0,4915

CI = 0,0519865895552343; CR = 5,84118983766677

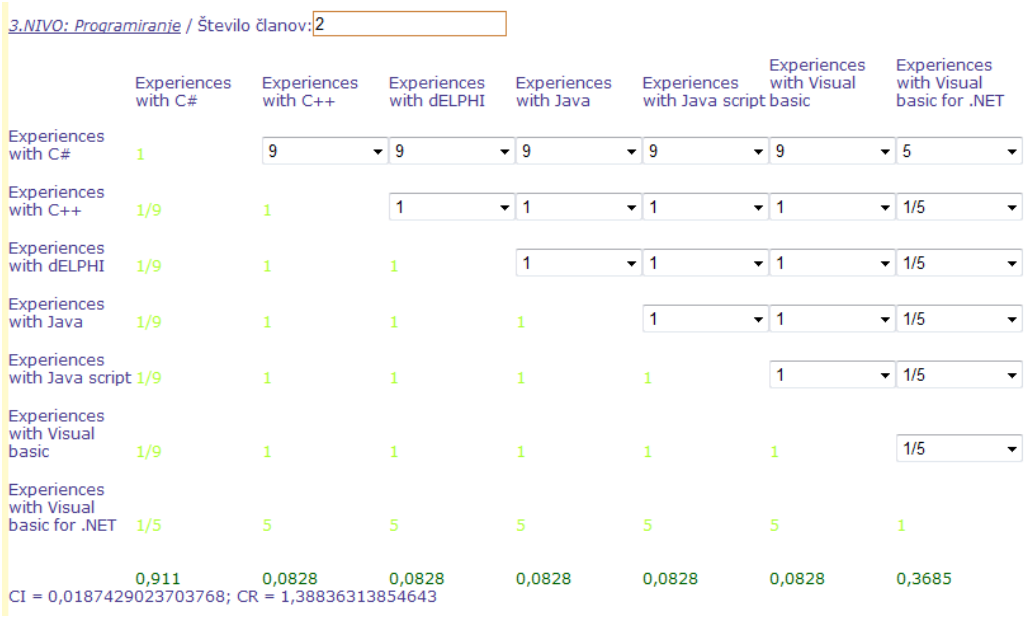
Želja je, da imata kandidata čimmanj izkušenj, ker se predvideva, da imajo manj izkušeni večjo željo po novitetah in hitrem učenju. Ocenimo, da je kandidat s 5-letnimi izkušnjami 4-krat vrednejši kot kandidat z 20-letnimi itd.

3.NIVO: Delovne izkušnje / Število članov: 2

	Do 10 let delovnih izkušenj	Do 20 let delovnih izkušenj	Do 30 let delovnih izkušenj	Več kot 30 let delovnih izkušenj	Do 5 let delovnih izkušenj
Do 10 let delovnih izkušenj	1	2	3	4	1/3
Do 20 let delovnih izkušenj	1/2	1	2	3	1/4
Do 30 let delovnih izkušenj	1/3	1/2	1	2	1/5
Več kot 30 let delovnih izkušenj	1/4	1/3	1/2	1	1/7
Do 5 let delovnih izkušenj	3	4	5	7	1
	0,3894	0,2421	0,1494	0,0919	0,8712

CI = 0,0191784042706007; CR = 1,72778416852258

Iščemo dva kandidata z znanjem C#. Ovrednotimo tudi znanje VB.NET, ker je tehnologija podobna in bi se kandidat lahko hitro prilagodil C#. Vseeno se da prednost znanju C# glede na VB.NET s količnikom 5.



3.NIVO: Programiranje / Število članov: 2

	Experiences with C#	Experiences with C++	Experiences with dELPHI	Experiences with Java	Experiences with Java script	Experiences with Visual basic	Experiences with Visual basic for .NET
Experiences with C#	1	9	9	9	9	9	5
Experiences with C++	1/9	1	1	1	1	1	1/5
Experiences with dELPHI	1/9	1	1	1	1	1	1/5
Experiences with Java	1/9	1	1	1	1	1	1/5
Experiences with Java script	1/9	1	1	1	1	1	1/5
Experiences with Visual basic	1/9	1	1	1	1	1	1/5
Experiences with Visual basic for .NET	1/5	5	5	5	5	5	1
	0,911	0,0828	0,0828	0,0828	0,0828	0,0828	0,3685

CI = 0,0187429023703768; CR = 1,38836313854643

Navedene zahteve izvedemo pri naslednjih kandidatih:

	delovne izkušnje	znanje c#	znanje VB.NET
a test	do 20 let	5	2
b test	do 20 let	4	1
c test	do 5 let	3	2
d test	do 5 let	1	5
e test	do 20 let	3	3

Ko se iskanje konča, lahko analiziramo rezultate:

	Delovne izkušnje (3 nivo)	Strokovna usposobljenost (2 nivo)
a test	0,108220	0,081550
b test	0,108220	0,081550
c test	0,389520	0,204340
d test	0,389520	0,204340
e test	0,108220	0,081550

	Skupna ocena (1 nivo)
a test	**0,207670**
b test	0,200840
c test	**0,202180**
d test	0,198350
e test	0,199910

	Programiranje (3 nivo)	Tehnično znanje (2 nivo)
a test	0,430950	0,224530
b test	0,334270	0,196730
c test	0,293220	0,184920
d test	0,238940	0,169310
e test	0,321040	0,192920

Izbrana sta kandidata a in c. Znanje kandidata a je prevladalo nad njegovimi delovnimi izkušnjami. Kandidat c je bil izbran zaradi manj delovnih izkušenj. Iz končnega koeficienta se opazi, da je kandidat c za malenkost premagal osebo b. Torej se lahko še vedno pretehta in raje vzamemo osebo z bogatejšim znanjem kot pa osebo z manj izkušnjami.

Primer 2:

Iščemo tim petih ljudi za celoten projekt. Rabimo tri programerje v Visual Studio programskem okolju, kjer bi programirali v VB.NET jeziku. Za dostop do podatkovne baze pa se bo uporabljal NHibernate. Tehnična dokumentacija bo deloma pisana v programu Visio, zato je zaželeno znanje le-tega. En član bi moral imeti dobro poznavanje administracije Oracle podatkovnih baz in njihovega dizajna, dva pa bi morala dobro poznati tudi spletne tehnologije, predvsem ASP.NET, in tudi spletno oblikovanje. Nekdo pa bo moral celoten projekt voditi in komunicirati s strankami. Od vseh članov pričakujemo dobro poznavanje angleškega jezika, saj bomo izdelovali programski paket za tujega naročnika. Vsaj od dveh kandidatov pričakujemo, da imata že večdesetletne delovne izkušnje in vsaj 7. stopnjo izobrazbe.

Za ta problem postavimo prednosti predvsem ocenam delovnih lastnosti in tehničnih znanj.

1.NIVO: Delovne in osebnostne lastnosti

	Delovne in osebnostne lastnosti	Splošni podatki	Strokovna usposobljenost	Tehnična znanja
Delovne in osebnostne lastnosti	1	3	3	1/3
Splošni podatki	1/3	1	1	1/5
Strokovna usposobljenost	1/3	1	1	1/5
Tehnična znanja	3	5	5	1
	0,396	0,1519	0,1519	0,8928

CI = 0,0150014450732498; CR = 1,68555562620784

Pri delovnih lastnostih močno izpostavimo vodenje, saj iščemo kanidata, ki ima dobre vodstvene lastnosti.

3.NIVO: Delovne lastnosti / Število članov: 1

	Gospodarnost	Kvaliteta dela	Storilnost	Vodenje
Gospodarnost	1	1	1/3	1/9
Kvaliteta dela	1	1	1	1/7
Storilnost	3	1	1	1/9
Vodenje	9	7	9	1
	0,0884	0,1206	0,1567	0,9763

CI = 0,0548982439660429; CR = 6,1683420186565

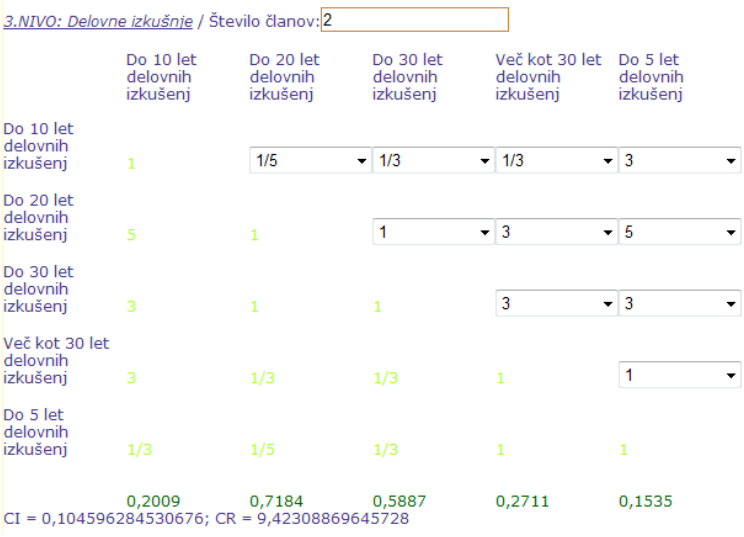
Za strokovno usposobljenost postavimo pogoj poznavanja tujega jezika na prvo mesto, podobno postavimo pogoj tudi za delovne izkušnje kandidata. Stopnja izobrazbe je manj pomemben pogoj.

2.NIVO: Strokovna usposobljenost

	Delovne izkušnje	Jeziki	Poklic	Smer izobrazbe	Stopnja izobrazbe
Delovne izkušnje	1	1	5	5	3
Jeziki	1	1	7	7	3
Poklic	1/5	1/7	1	1	1/5
Smer izobrazbe	1/5	1/7	1	1	1/5
Stopnja izobrazbe	1/3	1/3	5	5	1
	0,6252	0,6965	0,0919	0,0919	0,3273

CI = 0,038708394522317; CR = 3,48724274975829

Zaradi zahteve po dveh kandidatih z delovnimi izkušnjami postavimo prednost kandidatom z do 20 oz. z do 30 leti delovnih izkušenj. Več kot 30 let delovnih izkušenj ne izpostavimo kot dobro, saj je takrat lahko kandidat že starejši in izkušnje težko prepleta z modernejšo tehnologijo.



3.NIVO: Delovne izkušnje / Število članov: 2

	Do 10 let delovnih izkušenj	Do 20 let delovnih izkušenj	Do 30 let delovnih izkušenj	Več kot 30 let delovnih izkušenj	Do 5 let delovnih izkušenj
Do 10 let delovnih izkušenj	1	1/5	1/3	1/3	3
Do 20 let delovnih izkušenj	5	1	1	3	5
Do 30 let delovnih izkušenj	3	1	1	3	3
Več kot 30 let delovnih izkušenj	3	1/3	1/3	1	1
Do 5 let delovnih izkušenj	1/3	1/5	1/3	1	1
	0,2009	0,7184	0,5887	0,2711	0,1535

CI = 0,104596284530676; CR = 9,42308869645728

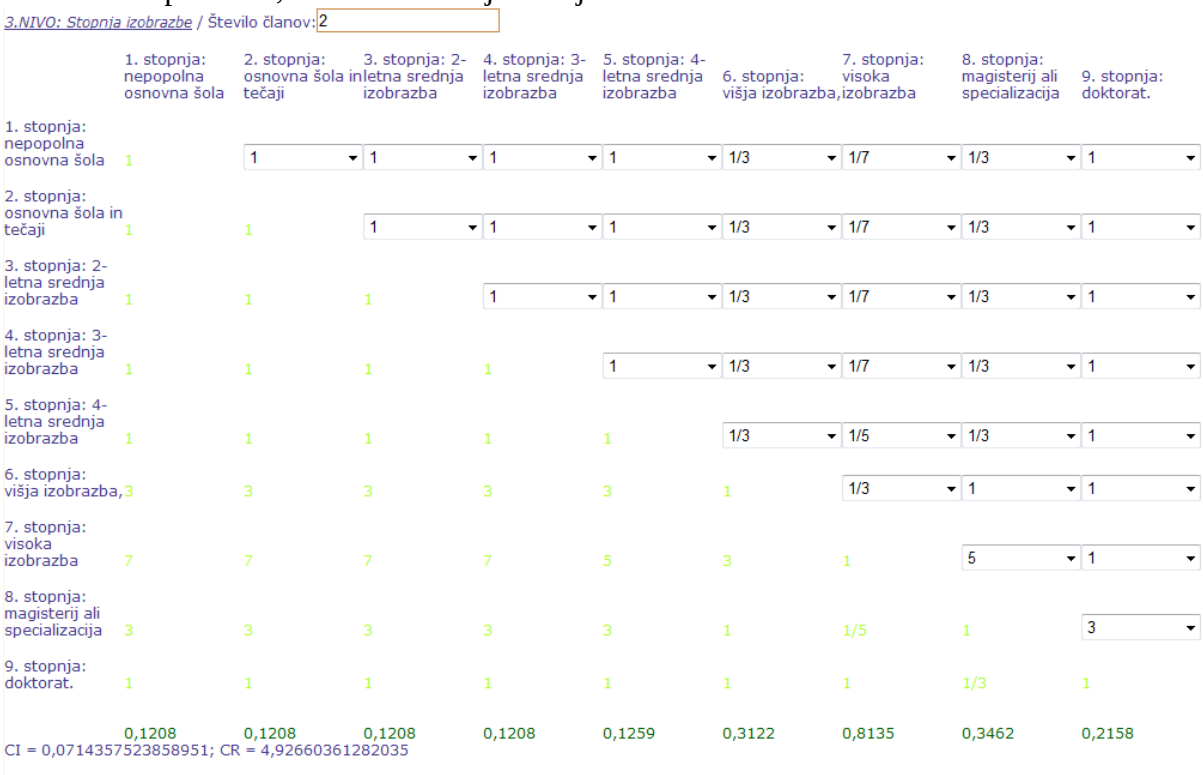
Za vse kandidate postavimo večji ocenjevalni faktor samo za poznavanje angleškega jezika.

3.NIVO: Jeziki / Število članov: 5

	Angleški jezik	Francoski jezik	Italjanski jezik	Nemški jezik	Ruski jezik
Angleški jezik	1	9	9	9	9
Francoski jezik	1/9	1	1	1	1
Italjanski jezik	1/9	1	1	1	1
Nemški jezik	1/9	1	1	1	1
Ruski jezik	1/9	1	1	1	1
	0,9762	0,1085	0,1085	0,1085	0,1085

CI = 0; CR = 0

Za stopnjo izobrazbe potrebujemo dva kandidata s 7. stopnjo izobrazbe, večja stopnja izobrazbe ni potrebna, zato tudi ni bolje ocenjena.



3.NIVO: Stopnja izobrazbe / Število članov: 2

	1. stopnja: nepopolna osnovna šola	2. stopnja: osnovna šola in tečaji	3. stopnja: 2-letna srednja izobrazba	4. stopnja: 3-letna srednja izobrazba	5. stopnja: 4-letna srednja izobrazba	6. stopnja: višja izobrazba,	7. stopnja: visoka izobrazba	8. stopnja: magisterij ali specializacija	9. stopnja: doktorat.
1. stopnja: nepopolna osnovna šola	1	1	1	1	1	1/3	1/7	1/3	1
2. stopnja: osnovna šola in tečaji	1	1	1	1	1	1/3	1/7	1/3	1
3. stopnja: 2-letna srednja izobrazba	1	1	1	1	1	1/3	1/7	1/3	1
4. stopnja: 3-letna srednja izobrazba	1	1	1	1	1	1/3	1/7	1/3	1
5. stopnja: 4-letna srednja izobrazba	1	1	1	1	1	1/3	1/5	1/3	1
6. stopnja: višja izobrazba,	3	3	3	3	3	1	1/3	1	1
7. stopnja: visoka izobrazba	7	7	7	7	5	3	1	5	1
8. stopnja: magisterij ali specializacija	3	3	3	3	3	1	1/5	1	3
9. stopnja: doktorat.	1	1	1	1	1	1	1	1/3	1
	0,1208	0,1208	0,1208	0,1208	0,1259	0,3122	0,8135	0,3462	0,2158

CI = 0,0714357523858951; CR = 4,92660361282035

Administratorja ne potrebujemo, potrebo po ostalih tehničnih znanjih pa ocenimo z istim faktorjem.

2.NIVO: Tehnična znanja

	Administracija	Baze podatkov	Programska orodja	Programiranje	Spletne tehnologije
Administracija	1	1/7	1/7	1/7	1/7
Baze podatkov	7	1	1	1	1
Programska orodja	7	1	1	1	1
Programiranje	7	1	1	1	1
Spletne tehnologije	7	1	1	1	1
	0,0698	0,4988	0,4988	0,4988	0,4988

CI = 0; CR = 0

Za kandidata s poznavanjem baze podatkov zahtevamo tudi, da je strokovnjak za izgradnjo in administracijo podatkovne baze. Dobro je, da ima kandidat vsaj nekaj izkušenj s poizvedbami. Ker se bo uporabljala Oracle podatkovna baza, je to glavni pogoj glede tipa.

3.NIVO: Baze podatkov / Število članov: 1

	Experiences with database design	Experiences with database programming	Experiences with database administration	Experiences with mySQL	Experiences with Oracle	Experiences with PostgresSQL	Experiences with SQL	Experiences with SQL Server	Experiences with UML	Experiences with XML
Experiences with database design	1	9	1	9	1	9	5	9	9	9
Experiences with database programming	1/9	1	1/9	1	1/9	1	1/5	1	1	1
Experiences with database administration	1	9	1	9	1	9	3	9	9	9
Experiences with mySQL	1/9	1	1/9	1	1/9	1	1/5	1	1	1
Experiences with Oracle	1	9	1	9	1	9	3	9	9	9
Experiences with PostgresSQL	1/9	1	1/9	1	1/9	1	1/5	1	1	1
Experiences with SQL	1/5	5	1/3	5	1/3	5	1	3	5	5
Experiences with SQL Server	1/9	1	1/9	1	1/9	1	1/3	1	1	1
Experiences with UML	1/9	1	1/9	1	1/9	1	1/5	1	1	1
Experiences with XML	1/9	1	1/9	1	1/9	1	1/5	1	1	1
	0,5854	0,0569	0,5407	0,0569	0,5412	0,057	0,2287	0,06	0,057	0,057

CI = 0,0113854424885434; CR = 0,764123656949224

Tako ocenimo tri kandidate, ki morajo obvladati orodje VS 2005. Vključimo še zaželeno znanje NHibernate knjižnice in MS Visio orodja.

3.NIVO: Programska orodja / Število članov: 3

	Experiences with Eclipse	Experiences with Microsoft Visio	Experiences with Net Beans	Experiences with NHibernate	Experiences with Nunit	Experiences with Oracle designer	Experiences with Oracle forms	Experiences with Visual studio 2005
Experiences with Eclipse	1	1/5	1	1/7	1	1	1	1/9
Experiences with Microsoft Visio	5	1	3	1/3	3	3	3	1/5
Experiences with Net Beans	1	1/3	1	1/7	1	1	1	1/9
Experiences with NHibernate	7	3	7	1	7	7	7	1/5
Experiences with Nunit	1	1/3	1	1/7	1	1	1	1/9
Experiences with Oracle designer	1	1/3	1	1/7	1	1	1	1/9
Experiences with Oracle forms	1	1/3	1	1/7	1	1	1	1/9
Experiences with Visual studio 2005	9	5	9	5	9	9	9	1
	0,0647	0,2013	0,068	0,4366	0,0682	0,0682	0,0682	0,8638

CI = 0,0345544990101393; CR = 2,46817850072424

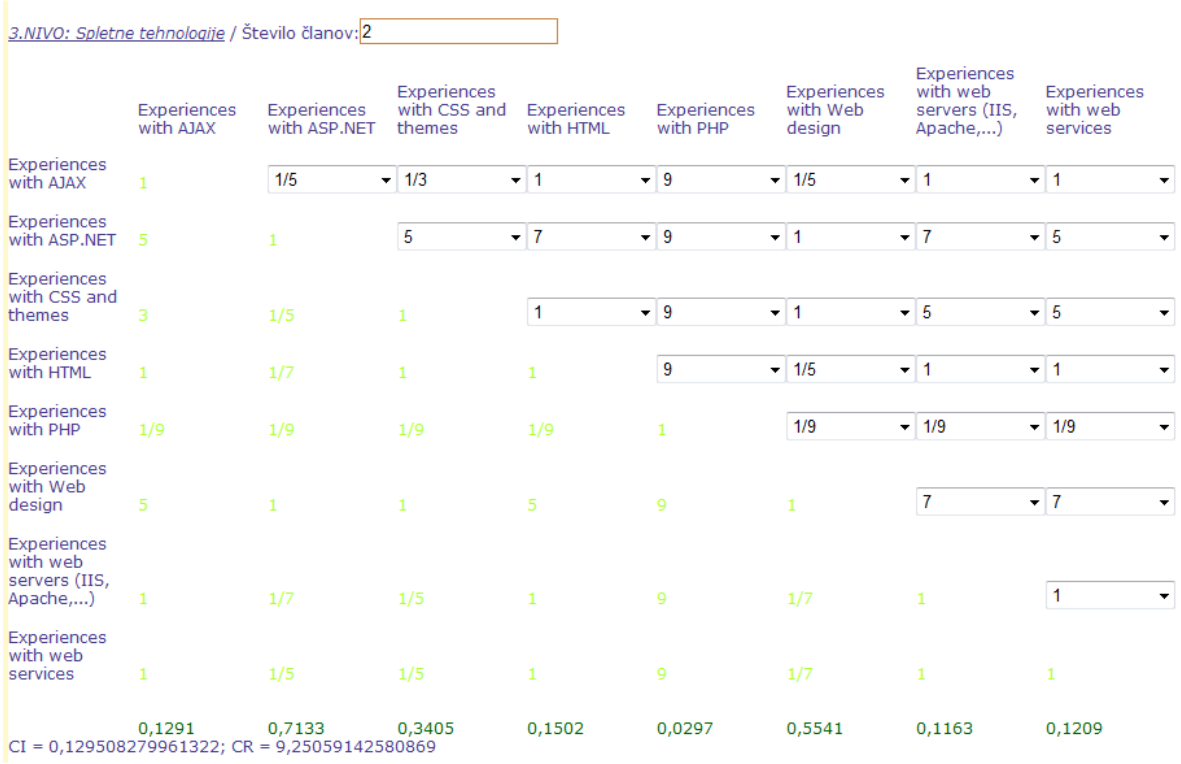
Za tri kandidate za programiranje v VB.NETu in JavaScriptu izberemo z naslednjo oceno pogojev:

3.NIVO: Programiranje / Število članov: 3

	Experiences with C#	Experiences with C++	Experiences with dELPHI	Experiences with Java	Experiences with Java script	Experiences with Visual basic	Experiences with Visual basic for .NET
Experiences with C#	1	1	1	1	1/3	1	1/9
Experiences with C++	1	1	1	1	1/3	1	1/9
Experiences with dELPHI	1	1	1	1	1/3	1	1/9
Experiences with Java	1	1	1	1	1/3	1	1/9
Experiences with Java script	3	3	3	3	1	3	1/3
Experiences with Visual basic	1	1	1	1	1/3	1	1/9
Experiences with Visual basic for .NET	9	9	9	9	3	9	1
	0,1015	0,1015	0,1015	0,1015	0,3068	0,1017	0,9243

CI = 2,7485612251373E-06; CR = 0,000203597127787948

Za spletne tehnologije morata imeti željena kandidata predvsem znanje o spletnem oblikovanju v okolju ASP.NET, najbolje s temami in CSS stili.



3.NIVO: Spletne tehnologije / Število članov: 2

	Experiences with AJAX	Experiences with ASP.NET	Experiences with CSS and themes	Experiences with HTML	Experiences with PHP	Experiences with Web design	Experiences with web servers (IIS, Apache,...)	Experiences with web services
Experiences with AJAX	1	1/5	1/3	1	9	1/5	1	1
Experiences with ASP.NET	5	1	5	7	9	1	7	5
Experiences with CSS and themes	3	1/5	1	1	9	1	5	5
Experiences with HTML	1	1/7	1	1	9	1/5	1	1
Experiences with PHP	1/9	1/9	1/9	1/9	1	1/9	1/9	1/9
Experiences with Web design	5	1	1	5	9	1	7	7
Experiences with web servers (IIS, Apache,...)	1	1/7	1/5	1	9	1/7	1	1
Experiences with web services	1	1/5	1/5	1	9	1/7	1	1
	0,1291	0,7133	0,3405	0,1502	0,0297	0,5541	0,1163	0,1209

CI = 0,129508279961322; CR = 9,25059142580869

Podane zahteve izvedemo pri naslednjih kandidatih, ki so se vpisali v portal. Z barvami označimo kandidate, ki izstopajo glede na določen atribut. Označene so vrednosti atributov: dobre vrednosti z rumeno in odlične z rdečo barvo. Glede na tako označevanje na hitro izberemo iz razpredelnice možne kandidate, da bomo na koncu lahko primerjali s kandidati, ki jih nam algoritem vrne.

	Odnos do delavcev	Odnos do dela	Odnos do strank	Vodenje	**Stopnja izobrazbe**	Angleški jezik	**Delovne izkušnje**
a test	1	1	4	4	7	6	do 5
b test	1	5	3	3	4	3	do 30
c test	1	9	4	2	7	6	do 20
d test	3	9	1	3	6	6	do 10
e test	3	3	9	7	5	5	do 5
f test	5	1	9	10	8	9	vec 30
g test	5	3	5	3	8	8	do 5
h test	5	3	5	10	7	3	do 20
i test	6	8	3	1	6	8	do 5
j test	6	1	8	3	6	3	do 30
k test	6	10	7	2	6	4	do 20
l test	6	1	6	8	6	6	vec 30
m test	7	5	10	9	9	10	do 20
n test	7	10	7	5	7	5	do 10
o test	7	1	9	8	5	8	do 10
p test	7	10	5	4	5	3	do 5
r test	8	5	10	9	7	7	do 30
s test	8	3	5	9	8	5	do 10
t test	10	2	9	6	6	7	do 20
u test	10	9	8	5	7	6	do 10

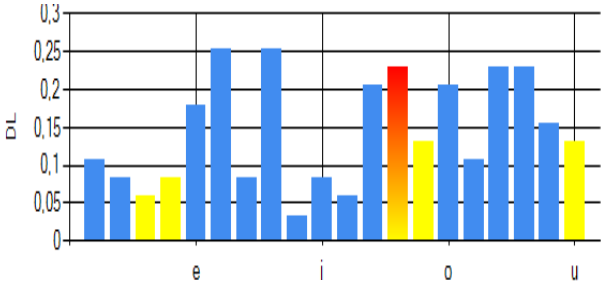
	VB. NET	Java script	VS 2005	MS Visio	NHibernate	ASP. NET	Web design	DB admin	DB design	Oracle
a test	5	2	5	1	2	3	3	1	1	1
b test	3	1	3	4	1	3	4	5	5	5
c test	4	4	4	2	3	5	2	2	4	2
d test	5	2	4	2	4	5	3	2	2	2
e test	2	2	2	2	1	2	3	1	1	2
f test	1	1	1	4	1	1	2	1	1	1
g test	2	3	2	2	1	2	5	1	2	1
h test	1	1	1	1	1	1	1	1	1	1
i test	3	1	3	3	3	3	2	3	3	3
j test	4	2	2	3	2	3	4	2	3	2
k test	3	5	3	2	1	4	4	2	4	4
l test	2	1	2	3	1	2	4	1	2	1
m test	1	1	1	1	1	1	2	1	1	2
n test	4	3	5	1	5	1	1	4	4	1
o test	5	2	5	1	1	1	1	3	3	3
p test	1	3	5	2	1	3	2	2	4	2
r test	1	1	1	2	1	1	4	1	1	1
s test	3	2	3	3	1	3	3	2	2	2
t test	3	4	3	2	2	2	2	2	2	2
u test	2	2	5	5	2	3	5	4	5	5

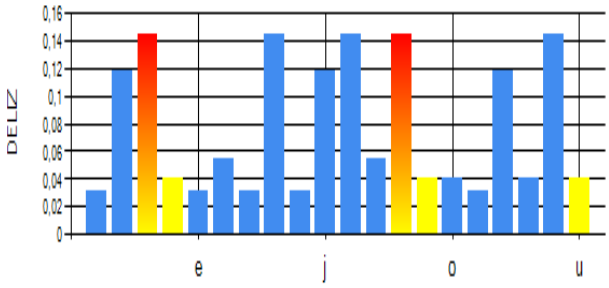
Ko se algoritem izvede nad tem naborom kandidatov, kot rezultat dobimo naslednje kandidate za posamezne atribute:

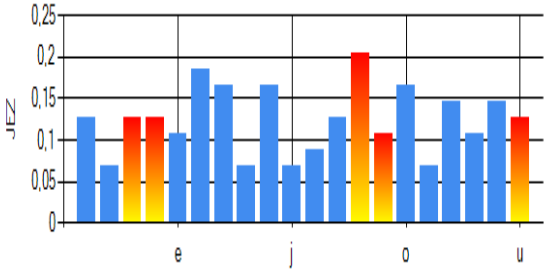
kanidat C:	programiranje,		izobrazba,	del. izkušnje
kandidat D:	programiranje,	orodja, web		
kandidat M:			vodenje,	del. izkušnje
kandidat N:	programiranje,	orodja		
kandidat U:		orodja, web,	izobrazba, DB,	

Pri analizi rezultatov so nam v pomoč grafi, ki jih algoritem izriše. Rumena barva na kandidatu pomeni izbran kandidat, rdeči zgornji del pa pomeni izbran kandidat za določen atribut. Kratice za atribute, po katerih se je ocenjevalo, so: DL – delovne lastnosti, DELIZ – delovne izkušnje, JEZ – jeziki, PROGR – programiranje, ORODJ – programska orodja, SPTE – spletne tehnologije, BAPO – baze podatkov, STIZ – stopnja izobrazbe.

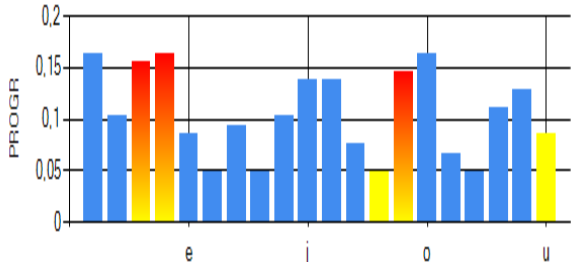
Za vodenje je bil izbran kandidat M, ki ima ravno prav delovnih izkušenj in dobro vodstveno oceno. Ima dober odnos do strank ter tudi dobro znanje angleškega jezika.

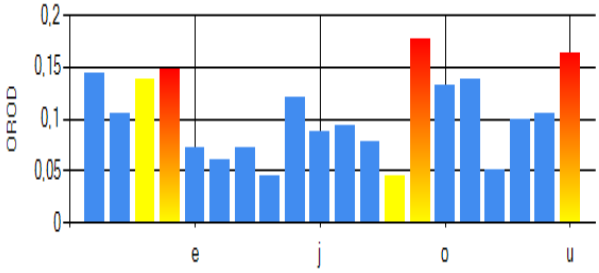


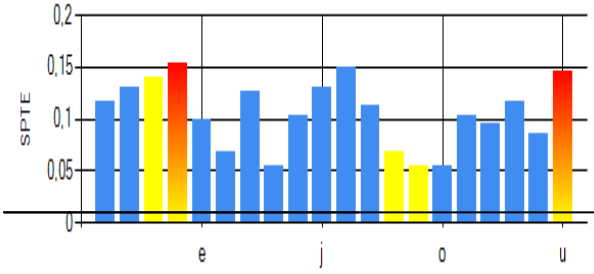




Za programiranje so bili izbrani kandidati C, D in N. Kandidat C ima dobre delovne izkušnje in zahtevano izobrazbo, čeprav ne pozna orodij, v katerih bi programirali, a se bo glede na izobrazbo in izkušnje to z lahkoto naučil. Kandidata D bi uporabili za programerja, ki obvlada tudi spletne tehnologije v okolju VS 2005.

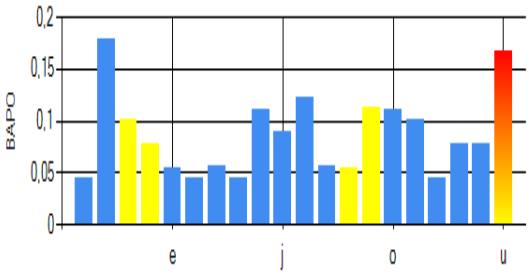


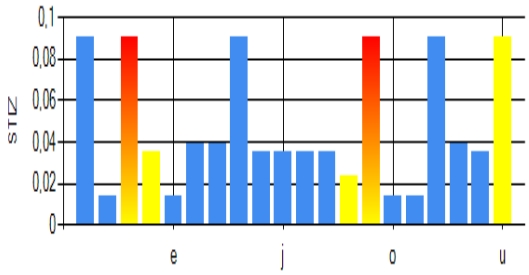




Kandidat N pa bo klasičen programer v okolju VS2005, ki bo opravljal tudi delo z NHibernatom.

Kandidat U bo drugi, ki se bo ukvarjal s spletno tehnologijo. Ker ne pozna orodja VS 2005, se bo ukvarjal bolj s spletnim oblikovanjem. Prav tako pa se bo ukvarjal tudi z Oracle bazo podatkov, saj ima dobro oceno iz znanja baz podatkov. Čeprav nima še veliko delovnih izkušenj, pa ima željeno stopnjo izobrazbe.





Za analizo rezultata so nam v veliko pomoč izrisani grafi, ki jih vrne algoritem. Ugotovimo lahko, da je rezultat v skladu s pričakovanji in da je bila izbrana najbolj optimalna rešitev. Vsi izbrani kandidati so bili v okviru kandidatov, ki smo jih na hitro označili z barvami na začetku primera. Poglobljena analiza pokaže smiselnost izbora kandidatov, ki jih vrne algoritem.

7 ZAKLJUČEK

V nalogi smo naredili analizo za portal in ga uspešno implementirali s pomočjo študentov. Izgradnjo portala smo razdelili na manjše ločene sklope, opisane v poglavju 3.3. Te sklope smo razdelili v izdelavo študentom glede na njihove sposobnosti, ki smo jih ugotovili skozi vprašalnik. Vprašalnik je bil sestavljen po sklopih in atributih, podanih v poglavju 2.3. Glede na to, da študentje niso imeli večjih vsebinskih težav pri izgradnji sklopov in so se gibali v okviru planiranega časa za izgradnjo, lahko rečemo, da je bila analiza ustrezno napisana.

V literaturi se pojavlja veliko večparametrskih modelov za odločanje glede izbire dobaviteljev, proizvajalcev in tudi kadra glede na zahtevane parametre. V poglavju 5 smo na podlagi teh modelov in teorije AHP in MAUT metod uspešno sestavili model za ocenjevanje zaželjenosti atributov posameznih kandidatov glede na potrebe projekta. Na podlagi teh ocen se s pomočjo optimizacijske metode celoštevilskega linearnega programiranja sestavi optimalen virtualni tim. Z dvema primeroma v poglavju 6 ponazorimo, katere rezultate model vrača in s tem tudi ponazorimo njegovo pravilnost.

Problematična je sama potratnost reševanja optimizacijskega problema. Reševanje celoštevilskega linearnega programiranja je NP poln problem, zato je pri velikem številu kandidatov zelo potratno. Algoritem, implementiran v portalu, v razumljivem času najde tim glede na okvirno 30 atributov izmed 200 kandidatov.

Iskanje bi lahko dodatno izpopolnili tako, da bi dodali še dodatne kriterije glede virtualizacije tima, poleg kraja bivanja še zahtevek o povprečni razdalji med dvema članoma.

V funkcionalnost portala bi lahko dodali dodatne funkcionalnosti o oceni kandidatov s strani iskalcev zaposlitve, tako bi dobili tudi povratno oceno opravljenega dela. Ta ocena bi potem izboljšala samo oceno kandidatov.

Osnovni cilj naloge smo dosegli, področje, ki smo ga odprli, pa nudi brezmejno število različnih priložnosti, sploh v času, ko so zelo popularna socialna omrežja. Kandidate bi lahko izbirali v teh omrežijih preko spletnih storitev, ki jih le-ta ponujajo. V takih primerih bi morali najti še dodatne možnosti omejevanja članov, saj uporabljen algoritem nikoli ne bo sposoben kreiranja timov na tako veliki množici članov.

PRILOGA A

Matrike povezav za vsako vozlišče v drevesu, ki je definirano v poglavju 5.2:

NIVO 1:

Ocena kandidata

	splošni podatki	strokovna usposobljenost	delo in osebne lastnosti	tehnični kriteriji
splošni podatki	1			
strokovna usposobljenost		1		
delo in osebne lastnosti			1	
tehnični kriteriji				1

NIVO 2:

Splošni podatki

	spol	starost	državljanstvo	cena
spol	1			
starost		1		
državljanstvo			1	
cena				1

NIVO 3:

Spol

	moški	ženska
moški	1	
ženska		1

NIVO 3:

Starost

	18-27	28-37	38-47	48-57	58-
18-27	1				
28-37		1			
38-47			1		
48-57				1	
58-					1

NIVO 3:

Državljanstvo

	slovensko	italijansko	hrvaško	avstrijsko	madžarsko
slovensko	1				
italijansko		1			
hrvaško			1		
avstrijsko				1	
madžarsko					1

NIVO 3:

Cena

	drag	srednje drag	normalna cena	srednje poceni	poceni

drag	1
srednje drag	1
normalna cena	1
srednje poceni	1
poceni	1

NIVO 2:

Strokovna usposobljenost

	stopnja izobrazbe	smer izobrazbe	poklic	tuji jezik	funkcionalna znanja	delovne izkušnje
stopnja izobrazbe	1					
smer izobrazbe		1				
poklic			1			
tuji jezik				1		
funkcionalna znanja					1	
delovne izkušnje						1

NIVO 3:

Stopnja izobrazbe

		I. stopnja	II. stopnja	III. stopnja	IV. stopnja	V. stopnja	VI. stopnja	VII. stopnja	VIII. stopnja	IX. stopnja
I.	stopnja	1								
II.	stopnja		1							
III.	stopnja			1						
IV.	stopnja				1					
V.	stopnja					1				
VI.	stopnja						1			

VII.	stopnja	1
VIII.	stopnja	1
IX.	stopnja	1

NIVO 3:

Smer izobrazbe

	računalniška	naravoslovna	elektrotehnična
računalniška	1		
naravoslovna		1	
elektrotehnična			1

NIVO 3:

Poklic

	enostavno delo			zahtevno delo
enostavno delo	1			
		1		
			1	
				1
zahtevno delo				1

NIVO 3:

Tuji jeziki

	angleški	nemški	italijanski	francoski	ruski
angleški	1				
nemški		1			
italijanski			1		

francoski	1
ruski	1

NIVO 3:

Delovne

izkušnje

	do 5	do 10	do 20	do 30	nad 30
do 5	1				
do 10		1			
do 20			1		
do 30				1	
nad 30					1

NIVO 2:

Delo in

osebne lastnosti

	storilnost	kvaliteta dela	odnos do dela	odnos do strank	odnos do sodelavcev	gospodarnost	vodenje	zdravstveno stanje
storilnost	1							
kvaliteta dela		1						
odnos do dela			1					
odnos do strank				1				
odnos do sodelavcev					1			
gospodarnost						1		
vodenje							1	
zdravstveno stanje								1

NIVO 2:

Tehnični kriteriji

	znanje programiranja	programerska orodja	baze podatkov	spletne tehnologije	administracija
znanje programiranja	1				
programerska orodja		1			
baze podatkov			1		
spletne tehnologije				1	
administracija					1

NIVO 3:

Znanje programiranja

	C++	C#	Visual basic	Visual basic for .NET	Delphi	Java script	Java
C++	1						
C#		1					
Visual basic			1				
Visual basic for .NET				1			
Delphi					1		
Java script						1	
Java							1

NIVO 3:

Programerska orodja

	Visual studio 2005	Net Beans	Eclipse	Oracle forms	Oracle designer	Microsoft Visio	NHibernate	Nunit
Visual studio 2005	1							
Net Beans		1						
Eclipse			1					
Oracle forms				1				
Oracle designer					1			
Microsoft Visio						1		
NHibernate							1	
Nunit								1

NIVO 3:

Baze podatkov

	database administration	database programming	database design	SQL	XML	UML
database administration	1					
database programming		1				
database design			1			
SQL				1		
XML					1	
UML						1

NIVO 3:

Spletne tehnologije

	ASP.NET	AJAX	PHP	web services	web servers (IIS, Apache,...)	HTML	web design	CSS, themes
ASP.NET	1							
AJAX		1						
PHP			1					
web services				1				
web servers (IIS, Apache,...)					1			
HTML						1		
web design							1	
CSS, themes								1

NIVO 3:

Administracija

	Windows	Linux	Network
Windows	1		
Linux		1	
Network			1

Viri in literatura

[1] ASP.NET. http://en.wikipedia.org/wiki/ASP.NET

[2] P. Baloh, *Izboljšanje standardne simpleks metode z algoritmom za zoženje tabele simpleksov in metodo potisni in povleci, Magistrsko delo*. Ljubljana, 2003.

[3] N. Bhushan and K. Rai, *Strategic Decision Making: Applying the Analytic Hierarchy Process*.: Springer-Verlag London Limited, 2004.

[4] M. Bohanec, *Odločanje in modeli*.: DMFA, 2006.

[5] J. Figueira, S. Greco, and M. Ehrgott, *Multiple Criteria Decision Analysis: State Of The Art Surveys*.: Springer Science + Business Media, Inc., 2005.

[6] G. Hertel, S. Geister, and Konradt U., "Managing virtual teams: A review of current empirical research," vol. 15, no. Human Resource Management Review, 2005.

[7] S. Hillier and J. Lieberman, "Introduction to operations research," in *Introduction to operations research*.: McGraw-Hill, 2001.

[8] E. Jereb, *Računalniško podprt večparametrski hierarhični model procesa izbire kadrov, Doktorska disertacija*. Kranj, 2000.

[9] E. Jereb, U. Rajkovic, and R. Rajkovič, "A Hierarchical Multi-Attribute System Approach to Personnel Selection," *International Journal of Selection and Assessment*, vol. 13, no. 3, pp. 198-205, september 2005.

[10] A. Kusiak, *Engeneering Design: Products, Processes, and Systems (Chapter 8: Team Formation)*. San Diego, CA: Academic Press, 1999.

[11] J.W. Lee and S.H. Kim, "An integrated approach for independent information system project selection," vol. 19, no. International Journal of Project Management, 2001.

[12] LPSolve. http://lpsolve.sourceforge.net/

[13] J. S. Lurey and M. S. Raisinghani, "An empirical study of best practices in virtual teams," vol. 38, no. Information & Management, 2001.

[14] Math.NET Project. http://mathnet.opensourcedotnet.info/

[15] Microsoft (2002), Application Architecture for.NET: Designing Applications and Services.

[16] Nhibernate. http://www.developer.com/net/asp/article.php/3709346

[17] T. L. Saaty, "Axiomatic Foundation of the Analytic Hierarchy Process," vol. 32, No.7, no. Management Science, 1986.

[18] D.Y. Sha and Z.H. Che, "Virtual integration with a multi-criteria partner selection model for the multi-echelon manufacturing system," vol. 25, no. Int J Adv Manuf Technol, 2005.

[19] SQL Server. http://en.wikipedia.org/wiki/Microsoft_SQL_Server

[20] D. Vavpotič, Š. Furlan, and M. Bajec, "A method for rapid creation of a virtual software development team," *Information systems development : challenges in practice, theory, and education*, vol. cop. 2009, no. 461-470, 2009.

[21] A.R. Ravindran VijayWadhwa, "Vendor selection in outsourcing," no. Computers & Operations research.

[22] R. M. Burton Wong, "Virtual Teams: What are their Characteristics, and Impact on Team Performance?," vol. 6, no. Computational & Mathematical Organization Theory, 2000.

[23] Z. Zhang, J. Lei, N. Cao, and K. Ng, "Evolution of Supplier Selection Criteria and Methods".