Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matej Romih

Spletna platforma za gradnjo hierarhičnih modelov pri odločitvenih problemih

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Aleksander Sadikov

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Študent naj prouči procese gradnje odločitvenih problemov na podlagi metodologije MCDA. Izdela naj platformo, ki mora podpirati potrebne funkcionalnosti za izvedbo gradnje odločitvenega modela pri uporabi procesov MCDA. Platforma naj omogoča vnos, uvoz in strukturiranje potrebnih podatkov, na podlagi katerih izvede tudi preračun v koristnosti ter uteževanje kriterijev. Platforma naj bo zasnovana za uporabo na čim večjem naboru naprav.

Za pomoč pri izdelavi diplomskega dela se zahvaljujem viš. pred. dr. Aleksandru Sadikovu in as. dr. Martinu Možini. Zahvaljujem se tudi Alešu Bokalu za sodelovanje pri izdelavi aplikacije. Zahvalil bi se tudi družini in prijateljem, ki so mi bili v času študija v podporo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
MCDA	Multi Criteria Decision Analysis	večparametrska odločitvena analiza
MACBETH	Measuring Attractiveness by Categorical Based Evaluation Technique	merjenje privlačnosti s kategorično tehniko vrednotenja
HTML	Hyper Text Markup Language	jezik za označevanje nadbesedila
CSS	Cascading Style Sheets	kaskadne stilske priloge
CPU	Central Processing Unit	centralna procesna enota
RAM	Random Access Memory	bralno-pisalni pomnilnik

Povzetek

Naslov: Spletna platforma za gradnjo hierarhičnih modelov pri odločitvenih problemih

Cilj diplomske naloge je razviti odprtokodno spletno platformo, namenjeno podpori izvedbe večparametrskega odločitvenega procesa. Platforma je neodvisna od uporabe operacijskega sistema. Celotna aplikacija se lahko razdeli na dva dela: del za gradnjo modela in del za analizo. Ta diplomska naloga zajema del gradnje modela. Vsebuje podrobnejši opis metod in postopkov gradnje modela ter opis izbora tehnologij in implementacije. Namen diplomskega dela je tudi izdelava uporabniškega priročnika za uporabo izdelane platforme z opisom korakov odločitvenega procesa in opisom v platformi implementiranih funkcionalnosti, ki podpirajo izvedbo odločitvenega procesa. Dodaten namen izdelave je razviti odprtokodno aplikacijo, ki bo bolj ustrezala procesu učenja pri predmetu odločitveni sistemi kot komercialni program, ki se je za ta namen uporabljal do zdaj.

Ključne besede: večparametrski odločitveni model, odločanje, odločitveno drevo, variante, odločitveni parametri, program za podporo odločanju, MACBETH.

Abstract

Title: A web platform for the construction of hierarchical models in decision-making problems

The aim of this thesis is to develop an open-source internet platform, intended as support of the multi-parameter decision-making process' realisation. The platform is independent of the use of an operating system. The application may be divided into two distinctions: the part of the model construction and the analytical part. This thesis encompasses the former distinction. It contains a detailed description of the methods and procedures of the model's construction, as well as a description of technological selection and implementation. The intent is also the composition of an instruction manual for use of the created platform, with a description of the decision-making process steps, and a description of the functions implemented into the platform that support the execution of the decision-making process. An additional purpose is the development of an open-source application that will better serve the learning process in the Decision-making Systems course, than the commercial programme currently in use.

Keywords: multiple-criteria decision model, decision, decision tree, options, criteria, decision support software, MACBETH

Poglavje 1

Uvod

Vsak izmed nas mora v življenju kar naprej sprejemati različne odločitve tako na osebnih kot na strokovnih področjih. Nekatere manjše odločitve sprejemamo s hitrim premislekom brez posebnega procesa odločanja. Pri manjših odločitvah se ponavadi odločimo na podlagi hitre presoje dobrih in slabih lastnosti, ki si jih lastijo posamezni možni izidi. Torej pri takih odločitvah ne uporabljamo strukturiranega procesa, ki bi nas vodil do kar se da razumne in nepristranske odločitve, saj to ne bi imelo smisla, ker bi za majhno odločitev, katere izid nima velikega vpliva, porabili precej časa. Velikokrat pa stojimo pred večjimi in težjimi odločitvami, ki jih ne moremo sprejeti hitro ter na tako preprost način. Na primer za odločanje o tem, kaj bomo jutri jedli za kosilo, bomo porabili bistveno manj časa kot pri odločitvi o nakupu novega avtomobila ali izboru fakultete, na katero se bomo vpisali. Pri kompleksnejših oz. težjih odločitvah si želimo priti do čim boljše končne odločitve, saj imajo te ponavadi za nas večji pomen, ker ima lahko njihov izid velik vpliv na naše življenje. Zato smo pripravljeni težjim odločitvam posvetiti več časa kot manjšim odločitvam.

Za doseg nepristranske, razumne in optimalne odločitve si lahko pomagamo z dobrim procesom odločanja. Med dobrim procesom odločanja problem bolje spoznamo, ga strukturiramo, ovrednotimo in analiziramo. Tako nam odločitveni proces omogoča, da na strukturiran način preko subjektiv-

nih presoj pridemo do dobrega izida odločitve, ki bo po vsej verjetnosti bolj koristen kot izid odločitve, sprejet s slabo definiranim procesom.

Skozi čas je bilo razvitih veliko odločitvenih metod, ki se med seboj razlikujejo po: vrsti potrebnih podatkov, različnih korakih procesa, načinu vrednotenja različnih odločitev, količini informacije, ki jo pridobimo, itd. To diplomsko delo je osredotočeno na odločitvene metode MCDA (Multi Criteria Decision Analysis).

Danes so nam lahko pri odločitvenih procesih v veliko pomoč različne programske opreme, ki služijo kot orodje, s katerimi na sistematičen in organiziran način lažje pridemo do bolj informirane ter boljše odločitve kot sicer. Programi nam omogočajo vnos podatkov in gradnjo odločitvenih modelov v začetnih korakih odločanja ter pomoč pri temeljitih analizah v zaključnih korakih odločitvenega procesa. Pri uporabi programja za namene odločanja se je treba zavedati, da nam služi kot podporno orodje, ki nam olajša delo in prihrani čas, kar pomeni, da programi za podporo pri odločanju ne sprejemajo odločitve namesto nas zgolj na podlagi vnesenih podatkov.

Trenutno ni lahko najti dobrih prosto dostopnih in odprtokodnih programskih orodij, namenjenih učenju odločitvenih metodologij s področja MCDA, kar je predstavljalo eno izmed glavnih motivacij za izdelavo aplikacije, saj smo študentje primorani uporabljati poskusno različico sicer komercialne programske opreme HiView3 [12]. Tako nas je pri nastajanju izdelka vodila predvsem misel na uporabo orodja v učne namene.

Odločitveni proces in ravno tako funkcionalnosti izdelane aplikacije je mogoče razdelili na dva ločena dela: gradnjo odločitvenega modela in analizo. Ta dela sta med seboj odvisna in sama po sebi ne nosita veliko koristi, saj nam zgrajeni model nič ne pomeni in ne prinaša nobene nove informacije, ki bi nam na kakršen koli način pripomogla pri odločitvi. Sama analiza pa je brez modela težko izvedljiva. Ta diplomska naloga obsega razvoj in opis dela spletne platforme, ki je namenjen gradnji večparametrskega odločitvenega modela. Preostali del razvite aplikacije, ki obsega korake analize v odločitvenem postopku, je nastal v okviru diplomskega dela Aleša Bokala [2].

Razvita aplikacija nam nudi pomoč pri odločanju z metodami s področja MCDA. Te metode si lastijo veliko dobrih lastnosti in se v praksi s pridom uporabljajo tako na gospodarskih kot političnih področjih. Za celovito in pravilno delovanje je bilo treba v aplikaciji podpreti funkcionalnosti, ki pri gradnji modela obsegajo: vnos in definiranje variant, med katerimi se odločamo, določanje in urejanje kriterijev, gradnjo odločitvenega drevesa v hierarhično strukturo kriterijev, uteževanje kriterijev v zgrajenem drevesu in določanje funkcij koristnosti različnih tipov. Potrebne funkcionalnosti za celovito analizo odločitve, ki so bile implementirane v okviru diplome [2], obsegajo različne vrste analiz, kot so: map analiza, analiza občutljivosti, kaj če analiza, maximin metoda, maximax metoda, manjvredne variante, leksikografska analiza in prispevek h koristnosti.

Poglavje 2

Večparametrsko odločanje

O večparametrskem odločanju govorimo, kadar imamo odločitveni problem, kjer se je treba odločiti med več možnostmi oz. variantami, ki so si med seboj sorodne in se jih da primerjati na podlagi parametrov (lastnosti, kriterijev), ki si jih lastijo. Gre za neke vrste problem izbire najbolj koristne variante iz množice variant, ki so nam na voljo. Taki odločitveni problemi v resničnem svetu zajemajo velik del odločitev, s katerimi se srečujemo. Naša (odločevalčeva) naloga je izbrati najbolj primerno oz. najbolj koristno varianto glede na naše osebne potrebe. Torej, če izbiramo nov avto, ne bomo nujno izbrali najboljšega avtomobila, ki nam ga prodajalci ponujajo, saj je ta verjetno za nas predrag ali pa nam razlika v ceni pomeni večjo vrednost kot razlika v ostalih lastnostih, ki bi jo za to razliko cene pridobili.

V tem poglavju je na kratko predstavljenih nekaj teoretičnih osnov, ki jih je koristno poznati pri izvedbi večparametrskega odločitvenega procesa.

2.1 Koristnost

S pojmom koristnosti pridobimo način za merjenje zaželenosti oz. preference posameznih variant z več kriteriji. Koristnost je številska vrednost oz. ocena, ki pripada določeni varianti in odločevalcu pove, katera varianta je boljša ter katera slabša. Koristnost odločevalcu omogoča možnost razporeda variant od

najbolj do najmanj zaželene, kjer velja tranzitivnost. Koristnost predstavlja osnovi pojem za analizo odločitvenega modela.

Končna koristnost variante se izračuna na podlagi **funkcije koristnosti**, ki ima nalogo pretvorbe variant oz. vrednosti njihovih kriterijev, ki so si med seboj različni, v poenoteno številsko vrednost oz. koristnost. Funkcija koristnosti to nalogo opravi v dveh korakih:

1. Prvi korak predstavlja izračun **delne funkcije koristnosti** (koristnosti posameznih kriterijev). Funkcije preslikajo vrednosti posameznega kriterija v vrednosti preference izbranega kriterija (stopnja zaželenosti v okviru naše odločitve). S prvim korakom se različni kriteriji postavijo na isti imenovalec, saj se vrednosti vsakega kriterija preslikajo v vrednosti med 0 in 100. To nam kasneje omogoča združitev koristnosti različnih parametrov v koristnost celotne variante.

2. Drugi korak združuje vrednosti, pridobljene v prvem koraku (za vsako varianto posebej). To je zaradi prvega koraka možno storiti z uteženo vsoto med pridobljenimi vrednostmi delnih koristnosti in utežmi, ki smo jih določili med procesom, kot je opisano kasneje 2.3.6. Pridobljene vrednosti variant predstavljajo njihove **končne koristnosti**.

Kot opisano zgoraj, se koristnosti izračunajo s podanim aditivnim modelom [1]:

$$\forall x_i \in X : U(x_i) = U(f_i(x_1), ..., f_q(x_i)) = \sum_{j=1}^{q} U_j(f_j(a_i)) * w_j \tag{2.1}$$

Kjer so:

X ..... množica variant $\{x_1, x_2, ..., x_n\}$
U ..... končna koristnost variante
U_j ..... funkcija koristnosti kriterija j
$f_j(a_i)$ ..... vrednost variante i za kriterij j
w_j ..... utež dodeljena kriteriju j

Vse uteži w_j so pri postopku normirane in zanje velja:

$$\sum_{j=1}^{q} w_j = 1$$

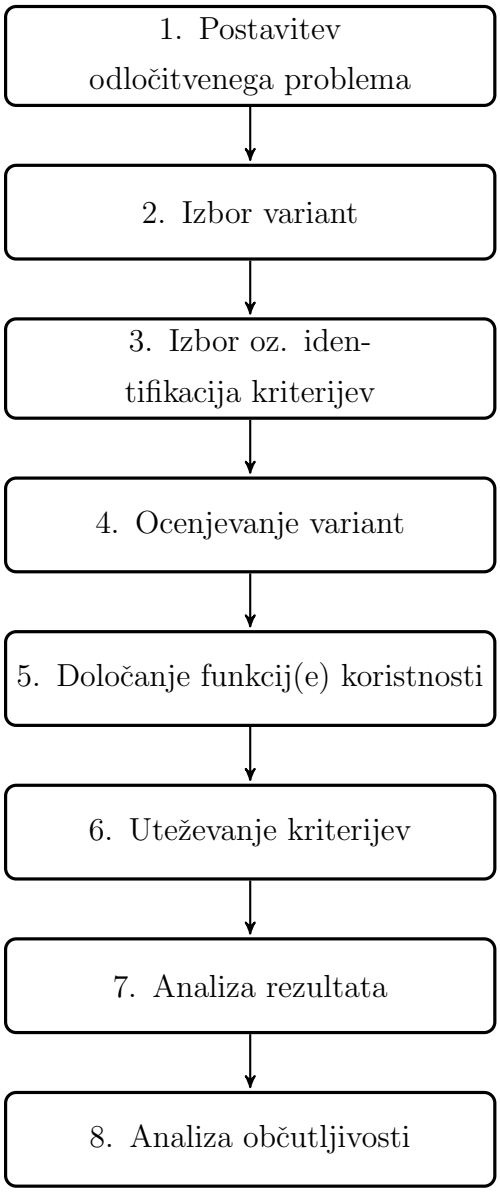
Primer izračuna koristnosti je opisan v podpoglavju 3.5.5.

2.2 Odločitveni proces

Izdelana aplikacija podpira izvedbo odločitvenega procesa, kjer sledimo korakom MCDA-metod, kot prikazuje slika 2.2. Kot že omenjeno, to diplomsko delo obsega gradnjo odločitvenega modela, ki zajema implementacijo potrebnih funkcionalnosti korakov 4, 5 in 6. Korak 1 ne potrebuje posebne podpore programa. Koraka 2 in 3 od aplikacije zahtevata zgolj vnos ter hranjenje izbranih kriterijev in variant. Z namenom celovitega pregleda in zaradi narave prepletanja gradnje modela ter analize tega sta v tem diplomskem delu na kratko opisana tudi koraka analize. Potrebne funkcionalnosti korakov 7 in 8 so bile implementirane v diplomi [2]. Omenjena diploma vsebuje tudi podrobnejši opis korakov analize odločitvenega procesa.

Koraki odločitvenega procesa so iterativni. Vrstni red nekaterih korakov je lahko pomešan, med samim procesom pa se lahko vrnemo na določen že opravljen korak. Tako je, ker skozi proces problem bolje spoznamo in pridemo do novih ugotovitev, ki zahtevajo vrnitev na prejšnje korake. Naprimer: pri koraku ocenjevanja variant ugotovimo, da imajo za neki kriterij vse variante iste vrednosti. To od nas zahteva odstranitev kriterija, torej vračanje v korak izbora kriterijev. Do vračanja lahko privedejo tudi spremembe naših (odločevalčevih) potreb oz. želja, do katerih pride med procesom odločanja.

Recimo, da imamo problem iskanja nove zaposlitve. Izberemo si nekaj podjetij, v katerih želimo delati in imajo odprta delovna mesta. Opravimo zaposlitvene razgovore in čakamo na odgovore. Po enem tednu dobimo več ponudb različnih podjetij. Zdaj se moramo odločiti za eno izmed delovnih mest, pri čemer ima vsako delovno mesto različne lastnosti (velikost podjetja,



začetna plača, pričakovana plača po enem letu, oddaljenost od doma, št. delovnih ur na teden ...). Začnemo z odločitvenim procesom, po dveh dneh pa nam iz novega podjetja sporočijo, da imajo tudi oni odprto delovno mesto. Zdaj se bomo morali vrniti in upoštevati novo pridobljene informacije, saj lahko te vplivajo na končno odločitev.

Prvi štirje koraki so del procesa, ki predstavlja strukturiranje problema in nam ponujajo [9]:

- boljše razumevanje problema,
- boljše razumevanje vrednosti, ki vplivajo na odločitev,
- osnovo za nadaljnjo analizo,
- skupen jezik za komunikacijo (predvsem pri skupinskem odločanju).

Cilj naslednjih dveh korakov je, da kot odločevalci ovrednotimo oz. ocenimo preference nad postavljenimi kriteriji. Cilji korakov analize so opazovanje in globlje razumevanje pridobljenih rezultatov ter ugotavljanje, kako spremembe v zgrajenem modelu vplivajo na odločitev in na tak način preverjajo trdnost naše končne odločitve.

2.3 Opis korakov odločitvenega procesa

2.3.1 Postavitev odločitvenega problema

Včasih se nam zdi, da je odločitveni problem sam po sebi dobro razumljiv in bi si želeli ta korak namenoma ali ne kar preskočiti. Vendar je tudi temu koraku vredno posvetiti nekaj pozornosti, saj je v resnici zelo pomemben. Razlogi za to se skrivajo v koristi, če problem bolje poznamo, preden ga začnemo reševati. Če storimo napako v tem koraku in si problem narobe sestavimo, se bodo nepravilnosti prenesle v nadaljnje korake, kar lahko privede tudi do popolnoma napačne odločitve. Torej je naša naloga s skrbno postavitvijo odločitvenega konteksta, da pridemo do reševanja pravega problema. Korak postavitve odločitvenega problema je še bolj bistven pri skupinskem odločanju, kjer imajo lahko različni člani različne interese in poglede ter je pravilno medsebojno razumevanje obvezno. Vsak član skupine tako dobi bolje definirano sliko s pogledi in potrebami ostalih članov, ki jih je treba pri odločitvi upoštevati.

V tem koraku se znajdemo, ko zaznamo, da imamo pred seboj odločitveno situacijo, za katero je treba postaviti pravilni odločitveni problem. Za dobro definiranje odločitvenega problema obstaja več različnih metod. Pri večini

metoda iščemo odgovore na vprašanja, kot so: kaj je naš ključni problem, kakšni so naši cilji, kaj si z odločitvijo želimo doseči, kaj nam je pomembno, kakšne možnosti imamo, kakšne so naše omejitve, kdo vse je vključen v odločanje, na koga bo imela odločitev vpliv, kakšna je vloga posameznega odločevalca itd. Cilja takih vprašanj sta priti do izbora pravilne metode odločanja, ki je primerna za reševanje našega problema, ter okvirni nabor primernih variant in lastnosti, ki so za rešitev našega problema pomembne.

Pri naboru korakov odločitvenega procesa, ki je opisan v tem delu, je vredno še omeniti, da se korak postavitve odločitvenega problema delno vsebinsko prekriva z naslednjima korakoma: izbor variant in identifikacija kriterijev. Vendar ta dva koraka ne definirata celotnega odločitvenega problema, kot je to v odločitvenem procesu potrebno.

2.3.2 Izbor variant

Pri marsikateri odločitvi imamo na možnost veliko množico variant. Za učinkovito nadaljevanje jih je potrebno izbrati le nekaj. Naprimer, če nakupujemo nov računalnik, imamo danes poplavo možnosti. Izberemo le nekaj računalnikov, ki so v okviru naših omejitev ter potreb in ki so pritegnili našo pozornost. Pazljivi moramo biti, da naš nabor variant vsebuje za nas najbolj zanimive variante saj, če bojo v našem naboru slabe variante bo takšna tudi naša končna odločitev ne glede na pazljivost pri nadaljnjih korakih.

Če imamo že definirane cilje in omejitve, potem si pri tem koraku lahko pomagamo na naslednje načine [9]:

- Z uporabo ciljev
  - Katera varianta je najbolj zaželena, če bi uporabili samo en cilj?
  - Katera varianta je najbolj zaželena, če bi uporabili dva cilja?
  - Nadaljevanje postopka, dokler nimamo vključenih vseh ciljev.
- Z odstranitvijo omejitev

  - Odstranitvijo omejitve na variantah oz. posledice mogoče tudi tvorijo alternativo.
  - Katera varianta bi bila najbolj zaželena, če cena ne bi bila upoštevana?

- Z uporabo različnih vidikov
  - Kaj bi bila najbolj zaželena varianta iz samo določenega (enega) odločevalčevega vidika?

2.3.3 Izbor oz. identifikacija kriterijev

Ko imamo ožji nabor variant, je treba narediti izbor kriterijev. Tudi pri tem koraku si lahko pomagamo s cilji, ki smo si jih zastavili v prvem koraku. Pri nakupu računalnika imamo veliko možnih kriterijev, vendar vseh ne moremo upoštevati. Če izbiramo računalnik, ki ga potrebujemo za delo s pisarniškimi orodji, bodo za nas pomembne druge lastnosti, kot če izbiramo računalnik za delo z grafično vsebino.

Različne metode MCDA zahtevajo različno sestavo, vsebino in način, kako so kriteriji pridobljeni. Kljub temu je pri vseh metodah treba upoštevati naslednje (povzeto po [10]):

- **Pomembnost vrednosti:** kriteriji morajo predstavljati prave vrednosti zadanih ciljev. Imeti moramo omogočeno določanje preference, ki je povezana s konceptom cilja.
- **Razumljivost:** pomembno je, da imamo odločevalci skupno razumevanje koncepta, uporabljenega pri analizi.
- **Merljivost:** za kriterije mora biti mogoča merljivost performance variant.
- **Neodvečnost:** kriteriji ne smejo biti odvečni. Če obstaja več kriterijev, ki merijo isti faktor, bo ta imel pri odločitvi večjo težo.

- **Odločevalna neodvisnost:** kriteriji so odločevalno odvisni, če je preferenca posameznega kriterija ali kompromis med dvema kriterijema odvisen od vrednosti drugega kriterija. Na primer: če je avto črne barve, imam raje znamko BMW, če pa je srebrne barve, imam raje avto znamke Volkswagen.

- **Ravnotežje popolnosti in zgoščenosti:** zaželeno je, da so kriteriji kar se da popolni in zajamejo vse pomembne vidike problema ter pri tem zadržijo podrobnosti kar se da minimalno.

- **Operativnost:** naj bo za uporabo modela potreben le razumen trud. Naj ne bo treba porabiti preveč virov za pridobitev informacij.

- **Enostavnost proti kompleksnosti:** iz izkušenj je dobro stremeti k enostavnosti odločitvenega drevesa, ki primerno zajame problem. V praksi so pogosto predstavitve v začetnih fazah bolj podrobne, kot je potrebno in operativno zaželeno.

Če imamo že izbrane kriterije in se odpravimo na izbor oz. izločanje variant, je pametno pred nadaljevanjem še enkrat preveriti kriterije, saj se lahko zgodi, da za našo odločitev sicer pomemben kriterij postane nepomemben. Na primer: v novem naboru variant imajo vse enako vrednost za določen kriterij, torej kriterij v resnici nima vpliva, model pa je zaradi njega manj pregleden.

V tem koraku lahko kriterijem določimo tudi drevesno strukturo. Koren drevesa predstavlja našo idealno odločitev. Veje, izpeljane iz korena (in vsakega vozlišča), predstavljajo skupino kriterijev, ki so si med seboj sorodni. Hierarhijo lahko gradimo s pomočjo postavljenih ciljev, kjer kriteriji v določeni veji vsebinsko predstavljajo merilo za doseg posameznega cilja. Kriteriji bližje korenu predstavljajo bolj splošne cilje, medtem ko listi predstavljajo konkretne kriterije, s katerimi lahko merimo, v kolikšni meri variante ta cilj dosegajo. Primer drevesa prikazuje slika 3.1. Pri zadanem problemu je bil eden izmed ciljev izbor kar se da zmogljive mobilne naprave. Tako

zmogljivost predstavlja svojo vejo, sestavljeno s kriterijema RAM in CPU. RAM je že izmerljiv kriterij, samo zmogljivost centralne procesne enote pa merimo s št. jeder in hitrostjo (taktom).

2.3.4 Ocenjevanje variant

Po dobro definiranem odločitvenem problemu in po končanem izboru kriterijev ter variant je treba začeti gradnjo modela v aplikaciji z vnosom izbranih podatkov. Cilj je vnesti podatke v taki obliki, ki jih lahko uporabimo kot vhod za izračun funkcije koristnosti.

Vrednosti variant za kriterije, ki imajo direktno merljive vrednosti (monetarne, velikost itd.), moramo pred vnosom pretvoriti v poenoteno mersko enoto. Tako se taki podatki lahko kasneje uporabijo v funkciji koristnosti. Nekateri kriteriji nam tega ne omogočajo, ker si lastijo podatke takega tipa, ki ga ni mogoče pretvoriti v direktno merljive vrednosti, ki bi jo funkcija koristnosti kriterija neposredno preslikala v koristnost. Tipični primer takega kriterija je privlačnost. Pri takih kriterijih bi lahko določili kategorije (manj privlačen, bolj privlačen itd.) in jih dodelili variantam. Kasneje bi pri določanju funkciji koristnosti posamezni kategoriji določili vrednost koristnosti, v katero se kategorija preslika. Vendar je navadno boljši pristop z uporabo metode MACBETH. Primer uporabe si lahko ogledate v poglavju 3.5.3. Iz teoretičnega vidika se v takem primeru varianta kot taka (in ne kot vrednost kriterija) preslika direktno v koristnost.

Ko imamo podatke vseh variant vnesene in so ti kasneje preslikani v koristnosti, je pametno storiti pregled variant ter odstraniti tiste, ki so nepotrebne, saj tako povečamo preglednost in upravljivost nastalega modela. Odstranimo lahko manj vredne variante, za katere velja, da obstaja neka druga varianta, ki je boljša v vseh kriterijih.

2.3.5 Določitev funkcije koristnosti

Za vse kriterije je treba določiti funkcije koristnosti, s katerimi pridobimo iz vrednosti variant lokalno koristnost kriterijev, ki so kasneje združene v končno koristnost variant. Funkcije koristnosti so različnih tipov. Tip funkcije je odvisen od tipa in vsebine podatkov kriterija:

- **Linearne** funkcije lahko pokrijejo podatke številskega tipa, kjer se najnižja vrednost preslika v koristnost 0 in najvišja v koristnost 1 (ali obratno). Vmesne vrednosti se v koristnost preslikajo s pomočjo linearne interpolacije. Linearni funkciji koristnosti je treba torej določiti najboljšo in najslabšo vrednost.

- **Odsekovne** funkcije je treba uporabiti pri številskih podatkih, kadar nam koristnost ne narašča oz. pada linearno na celotnem intervalu glede na vrednosti kriterija. Odsekovna funkcija je sestavljena iz več manjših linearnih odsekov, ki jih moramo določiti sami glede na svoje potrebe. Z odsekovno funkcijo si lahko pomagamo tudi v primeru, kadar imamo potrebo po nelinearni funkciji, tako da z njenimi odseki poskusimo aproksimirati zaželeno funkcijo.

- **Diskretne** funkcije pa običajno uporabimo pri neštevilskih vrednostih oz. kadar ima parameter diskretne (nominalne) vrednosti. Funkciji določimo kategorije (posamezne vrednosti parametra) in vsaki kategoriji dodelimo vrednost koristnosti, v katero se preslika. Delo si lahko olajšamo tako, da najboljši vrednosti določimo koristnost 100, najslabši pa 0. Pri določanju vmesnih vrednosti moramo paziti, da relativna razlika med kategorijami določa razliko moči preference med njimi. Včasih pa imamo diskretni parameter, ki mu težko izmerimo vrednost oz. nima enote, s katero bi si lahko pomagali pri določanju njegove funkcije koristnosti. Za take primere lahko uporabimo metodo MACBETH, ki je podrobneje opisana kasneje v poglavju 5.

Skale vseh tipov funkcij so lahko relativne, kar pomeni, da je funkcija definirana na intervalu od najmanjše do največje vrednosti, ki je v naboru variant

za posamezni kriterij. Linearne in odsekovne funkcije pa dopuščajo tudi določitev fiksne skale. V tem primeru lahko določimo interval, na katerem je funkcija definirana. Vrednosti variant fiksnega kriterija ne smejo prekoračiti mej določenega intervala.

2.3.6 Uteževanje

Uteževanje kriterijev je zelo pomemben korak, saj ima določanje uteži neposreden vpliv na končno izbiro. Z uteževanjem določamo moč vpliva posameznih kriterijev in na tak način iščemo oz. postavljamo kompromise med njimi. V praksi obstaja več načinov uteževanja odločitvenega drevesa. V aplikaciji se uporablja hierarhično določevanje uteži, kjer določimo uteži na vsakem nivoju odločitvenega drevesa. Končna utež kriterija se izračuna z množenjem po nivoju normaliziranih vrednosti uteži na poti od korena do kriterija. Primer takega izračuna je podan spodaj. Funkcija koristnosti celotne variante potrebuje uteži končnih atributov oz. uteži listov drevesa.

Pri uteževanju moramo biti pozorni na razpone (intervale) vrednosti kriterijev, ki ga ocenjujemo. Na primer: pri izboru novega avtomobila je eden izmed ciljev odločitve minimizacija stroškov nakupa. Tako bi si sprva mislili, da ima cena pri odločitvi zelo pomembno vlogo. Med procesom odločanja je bil izbran ožji nabor petih variant, ki je vnesen v model. Lahko se zgodi, da je v naboru cena najcenejšega avtomobila 25.000 €, najdražjega pa 25.500 €. Torej je razlika med najdražjim in najcenejšim avtomobilom 500 €, kar pomeni, da mogoče cena le ne bo utežena kot zelo pomemben kriterij. Zavedati se je treba, da v resnici utežujemo pomembnost razlik v razponih kriterijev in tudi temu primerno postavimo uteži.

Uteževanje lahko začnemo s pomočjo naslednjega vprašanja: "Kateri razpon vrednosti kriterija izmed vseh, ki jih utežujemo, nam predstavlja največjo razliko?" Tako izbrani parameter utežimo z vrednostjo 100. Ostale kriterije lahko utežimo sorazmerno nanj, torej iščemo kompromis med najpomembnejšim kriterijem in trenutnim kriterijem. Pri uteževanju ostalih kriterijev si lahko pomagamo z vprašanjem: "Koliko najpomembnejšega kriterija

sem pripravljen žrtvovati za vrednost razpona kriterija, ki ga utežujemo?"

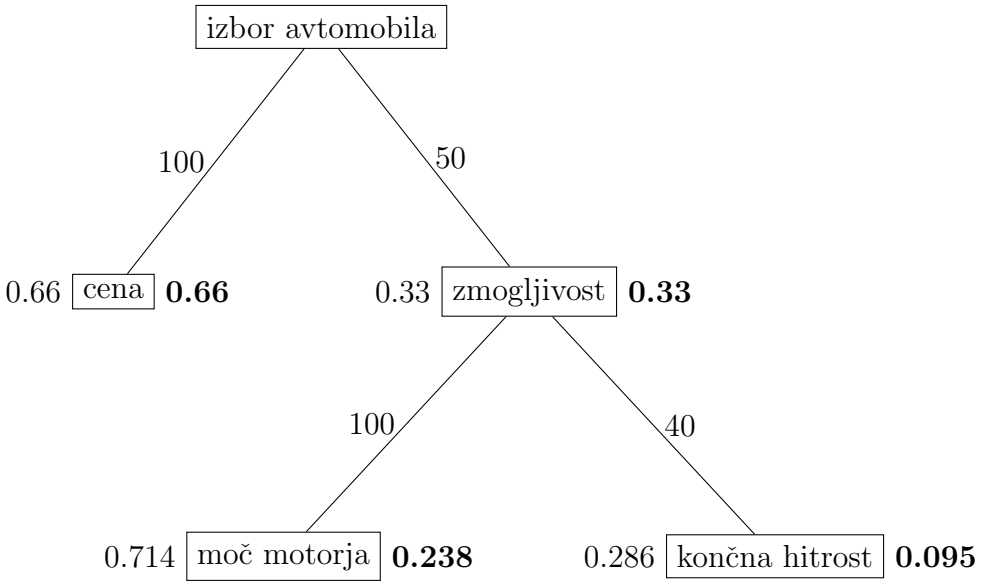
Recimo, da utežujemo kriterija cena in moč motorja. Razpon cene je 500 €, razpon moči motorja pa 77 kW. Ceno kot najpomembnejšo razliko ocenimo z utežjo 100. Zdaj se vprašamo, koliko cene smo pripravljeni žrtvovati za 77 kW. Če ocenimo moč motorja z utežjo 50, to pomeni, da nam je moč motorja 77 kW za polovico manj pomembna kot 500 €, ali drugače: 250 € nam pomeni enako kot 77 kW.

Na sliki 2.3.6 je prikazan manjši primer uteženega odločitvenega drevesa. Uteži ob povezavah predstavljajo uteži, ki so določene s strani odločevalca. Ob vsakem vozlišču je na levi prikaz **relativne uteži**, ki predstavlja normalizirano vrednost uteži kriterija glede na ostale kriterije iz iste družine drevesa. Na desni strani vozlišča pa je v krepki pisavi **končna utež** kriterija. Podana sta tudi postopek in izračun končnih uteži kriterijev, kot ga uporablja izdelana aplikacija:

- **cena:** $\frac{100}{150} = \frac{2}{3} = 0,\overline{66}$
- **zmogljivost:** $\frac{50}{150} = \frac{1}{3} = 0,\overline{33}$
- **moč motorja:** $\frac{50}{150} * \frac{100}{140} = \frac{1}{3} * \frac{5}{7} = \frac{5}{21} = 0,238$
- **končna hitrost:** $\frac{50}{150} * \frac{40}{140} = \frac{1}{3} * \frac{2}{7} = \frac{2}{21} = 0,095$

Uteži listov se, kot je treba, seštejejo v 1: $0,\overline{66} + 0,2\overline{38} + 0,095 = 1$. Končne uteži vozlišč imajo enako vrednost kot vsota končnih uteži otrok.

Eden izmed načinov uteževanja je od spodaj navzgor, kjer najprej utežujemo liste posamezne družine na spodnjem nivoju drevesa (po postopku opisanem zgoraj). Ko se skozi postopek dvigujemo na višje nivoje, moramo za pravilno interpretacijo uteži primerjati razpone vozlišč, ki pa nimajo razpona, saj to niso konkretni kriteriji. Zato na višje nivoje za primerjavo vozlišča s seboj odnesemo najtežji kriterij celotne njegove družine. Tako je v zgornjem primeru bila sprva kot najpomembnejši kriterij ocenjena moč motorja. V drugem koraku, ko sta se primerjali cena in zmogljivost, je bilo določeno, da



Slika 2.1: Primer uteženega odločitvenega drevesa

je cena dvakrat bolj utežena kot moč motorja, ki je najpomembnejši kriterij zmogljivosti.

2.3.7 Analiza rezultata

Z zgrajenim modelom smo pridobili rezultate, ki nosijo informacije o koristnosti variante. Vendar pregled koristnosti posameznih variant in izbor najbolje ocenjene variante ne predstavlja temeljito opravljene odločitve. Z variantami in njihovimi koristnostmi se moramo malo od bližje spoznati. Med analizo rezultatov se poglobimo v vprašanja kot: "Zakaj je neka varianta tako koristna, kot jo prikaže izračun koristnosti?" Ugotavljamo tudi, kakšen vpliv imajo posamezni kriteriji pri doseganju rezultata za posamezne variante, opazujemo, kako se koristnost spreminja s spreminjanjem vrednosti kriterijev, itd. Pri analizi rezultata se uporabljajo metode: minimax, maximin, map analiza, leksikografska analiza. Take metode podrobneje opisuje [2].

Na tem mestu se splača preveriti tudi, ali smo pri gradnji na kaj pozabili.

Mogoče so bile prisotne pristranosti. Ali smo padli v katero izmed pasti, ki se pri odločitvah rade pojavijo? Več o pogostih pasteh pri odločitvenih procesih je možno prebrati v članku [7].

2.3.8 Analiza občutljivosti

Pri tej analizi testiramo trdnost našega modela oz. njegovo odpornost proti spremembam. Predvsem se osredotočimo na spremembo nastavljenih uteži. Pod drobnogled vzamemo utež izbranega kriterija in poskušamo ugotoviti, kaj bi se zgodilo, če bi imel ta kriterij manjšo ali večjo utež. Kako velika sprememba uteži določenega kriterija je potrebna, da pri odločitvi prevlada druga varianta? S takimi ugotovitvami poskušamo dobiti občutek za trenutno nastavljene uteži, nakar razmislimo, ali se je treba vrniti na uteževanje in postopek ponoviti. Najbolj uporabni metodi analize občutljivosti sta: metoda občutljivosti navzgor in metoda občutljivosti navzdol. Metode in proces analize občutljivosti so ravno tako podrobneje opisane v delu [2].

Poglavje 3

Gradnja odločitvenega modela

V tem poglavju je podrobneje opisana gradnja odločitvenega modela z uporabo izdelane aplikacije. Na začetku je predstavljen odločitveni primer oz. problem, ki se za razlago uporablja čez celotno poglavje. Podrobnejši opis uporabe aplikacije pri analizi je v že omenjenem diplomskem delu [2].

3.1 Opis odločitvene situacije in določitvenega problema

Recimo, da se nam je pokvaril mobilni telefon. Zdaj imamo pred seboj odločitveno situacijo, v kateri moramo izbrati novo napravo. Ker nimamo veliko denarja in si želimo dobiti dober telefon, smo pripravljeni za kakovostnejšo odločitev v odločitveni proces investirati nekaj časa. Sprva nam pade na pamet vprašanje, kateri telefon kupiti. Vendar si lahko zastavimo tudi širši problem. Mogoče imamo željo po minimizaciji stroškov z menjavo operaterja ali pa bi izbrali drugačen paket pri istem operaterju, saj nam je storitev pri njem všeč? Ali so pomembni samo naši pogledi, kaj pa družinski paketi? Če zamenjamo operaterja in imamo večino opravljenih pogovorov z ostalimi člani družine, katerih strošek bi se dvignil zaradi zamenjave operaterja, je mogoče vredno upoštevati tudi njihove potrebe. Seveda se je treba vprašati tudi, kaj pričakujemo od same naprave, saj imamo lahko danes drugačne

potrebe kot pri nakupu prejšnjega telefona. Nujno si je treba zastaviti točno definirane cilje, ki pa naj bodo v našem primeru:

- minimizacija stroškov,
- telefon naj bo čim bolj primeren za brskanje po spletu,
- zaradi narave našega dela naj bo telefon čim bolj primeren terenskemu delu,
- maksimiziranje zmogljivosti, zaradi uporabe procesorsko zahtevne aplikacije pri našem delu,
- pomemben nam je tudi videz in si želimo čim lepši telefon.

Kar pri svojem delu potrebujemo specifične aplikacije, ki so podprte zgolj na Androidnih sistemih na napravah z zaslonom na dotik, si postavimo tudi omejitvi glede izbora operacijskega sistema in vrste naprave. Pred vsako odločitvijo s stroški si je modro postaviti tudi cenovne omejitve, ki jih ne bomo presegli. V našem primeru smo pripravljeni za telefon odšteti maksimalno 650€. Spodnjo mejo pri ceni si postavimo na 400€.

Ker je naš ponudnik hitro in dobro razrešil težave v preteklosti ter ima dobro pokritost s signalom, smo odločeni, da ponudnika ne bomo menjali. Ker želimo doseči čim boljšo uporabniško izkušnjo pri uporabi interneta, je telefon, ki podpira samo protokol Edge, za nas neprimeren.

3.2 Izbor kriterijev in gradnja drevesa

V našem primeru bomo nadaljevali z identifikacijo kriterijev. Na izbor mobilnih naprav bomo prešli kasneje. Tak pristop VFT (Value focus thinking), kjer začnemo z razmislekom o vrednostih (kriterijih, lastnostih) zagovarja Kenny v svojem delu [8]. Glavna prednost glede na alternativni pristop AFT (Alternative focus thinking), kjer se najprej posvetimo variantam, je v tem, da imamo večjo kontrolo nad odločitveno situacijo. Vrednosti so tiste,

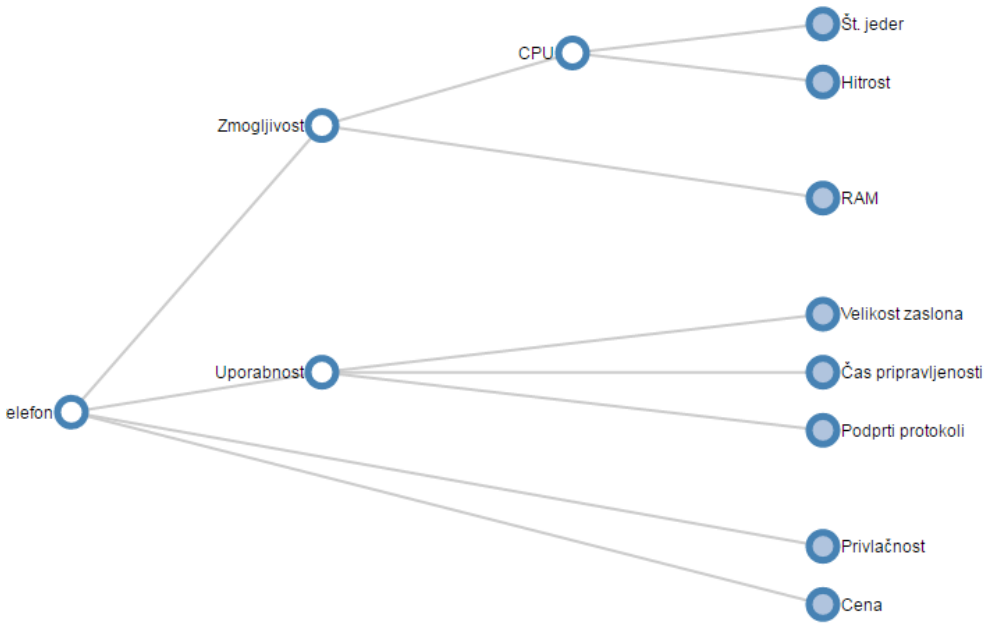
ki so temeljne pri odločanju in ne variante. V praksi se splača iterirati med obema pristopoma, a naj bodo vrednosti na prvem mestu.

Ob prvem pogledu na preglednico lastnosti mobilnih naprav se nam prikaže čez trideset možnih kriterijev. Za izbor potrebnih si pomagamo s cilji tako, da izberemo kriterije, ki se nam zdijo najbolj primerni za merjenje doseganja posameznega cilja. V našem primeru je za minimizacijo stroškov izbran kriterij **cena nakupa** z mersko enoto €. Za brskanje po spletu se nam zdita primerna atributa **velikost zaslona** merjen v milimetrih in podprti **internetni protokoli** (našteti po preferenci padajoče: LTE, HSPA, HSDPA, Edge). Za telefon, ki je primeren za terensko delo, štejemo telefon z vzdržljivo baterijo, kjer nam je bolj všeč telefon, ki ima večji **čas pripravljenosti** v minutah. Zmogljivost telefonov bo merjena v **velikosti pomnilnika RAM** in **številu jeder** ter **hitrosti procesorja**. Ker danes vsi telefoni podpirajo razširitev notranjega pomnilnika s spominsko kartico smo notranji pomnilnik iz nabora kriterijev izločili in na tak način povečali enostavnost modela. V modelu bo izgled telefona predstavljal kriterij **Privlačnost** brez merske enote - pomembna bo samo koristnost privlačnosti, ki jo bomo določili vsaki varianti.

Kriterije zgradimo v smiselne sklope in tvorimo hierarhično odločitveno drevo, kot prikazuje slika 3.1. Cilja glede primernosti telefona za terensko delo in uporabo interneta sta združena v vozlišče uporabnost. Maksimiziranje zmogljivosti predstavlja kriterij zmogljivost, ki je razdeljen na kriterij RAM in zmogljivost centralne procesne enote.

Pri urejanju kriterijev in gradnji drevesa se v aplikaciji srečamo s krožnim menijem, katerega ikone imajo naslednji pomen:

- kreiranje vmesnega vozlišča,
- kreiranje kriterija,
- odstranitev kriterija,
- lastnosti kriterija,



Slika 3.1: Prikaz primera odločitvenega drevesa.

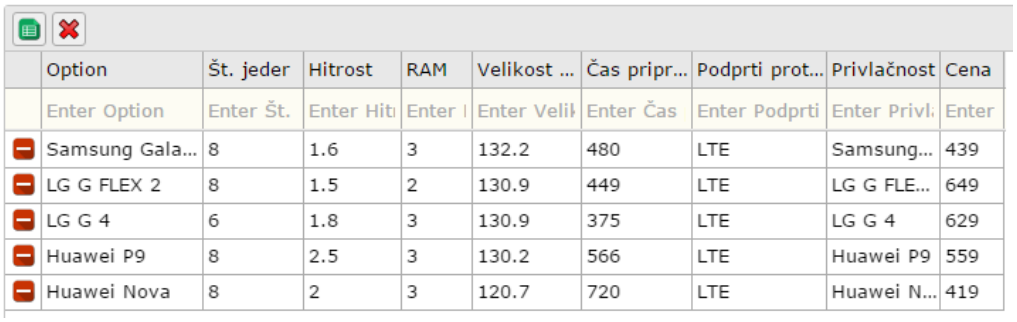
- uteževanje kriterijev pod vozliščem,
- urejanje funkcije koristnosti kriterija.

3.3 Izbor mobilnih naprav

Do sedaj smo si že izdelali približno sliko, kakšen naj bi bil naš novi telefon. V našem primeru imamo veliko možnih variant, zato jih sprva filtriramo po naših omejitvah, saj bomo s tem korakom močno skrčili nabor. Slika 3.2 prikazuje telefone na dotik z operacijskim sistemom Andorid in ceno med 400€ in 650€, ki so bili izbrani v ožji nabor variant.

Pri kriteriju podprti protokoli lahko opazimo, da imajo vsi telefoni enako vrednost - LTE. Tak kriterij je redundanten in ga odstranimo ter povečamo preglednost in enostavnost modela.

Pri vnosu variant aplikacija podpira uvoz podatkov iz tabelarnih datotek



Option	Št. jeder	Hitrost	RAM	Velikost ...	Čas pripr...	Podprti prot...	Privlačnost	Cena
Enter Option	Enter Št.	Enter Hit	Enter	Enter Veli	Enter Čas	Enter Podprti	Enter Privl.	Enter
Samsung Gala...	8	1.6	3	132.2	480	LTE	Samsung...	439
LG G FLEX 2	8	1.5	2	130.9	449	LTE	LG G FLE...	649
LG G 4	6	1.8	3	130.9	375	LTE	LG G 4	629
Huawei P9	8	2.5	3	130.2	566	LTE	Huawei P9	559
Huawei Nova	8	2	3	120.7	720	LTE	Huawei N...	419

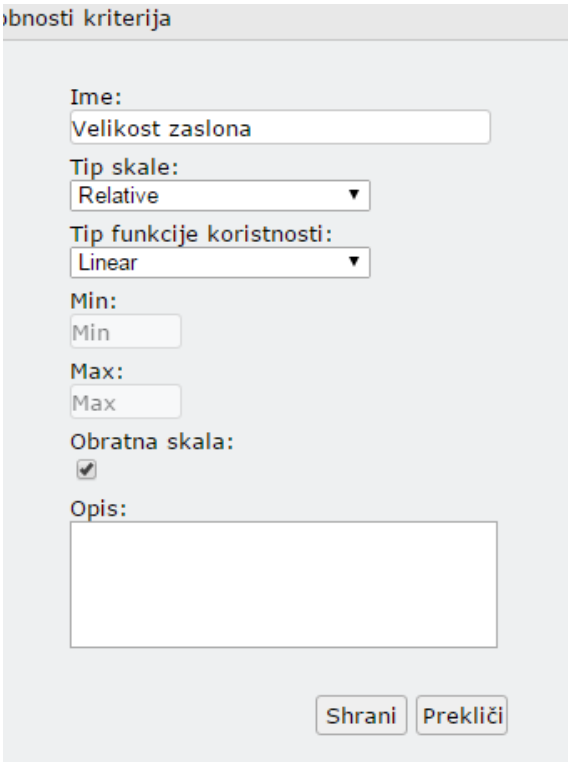
Slika 3.2: Prikaz ožjega nabora variant.

tipa .xls in .xlsx. To storimo s klikom na ikono in izborom datoteke. Pri tem mora biti vsebina strukturirana v obliki tabele, kjer prva vrstica predstavlja naslovno vrstico z imeni kriterijev. Pomembno je, da so imena kriterijev v datoteki enaka kot tista v aplikaciji, sicer podatki kriterija ne bodo uvoženi. Prvi stolpec je obvezen in mora biti poimenovan Option. Če pride pri uvozu do neskladnosti podatkov, se posamezne celice v aplikaciji obarvajo rdeče.

3.4 Urejanje kriterijev

Kriterijem določamo lastnosti s klikom na list drevesa, po prikazu krožnega menija pa kliknemo ikono . Prikaže se pojavno okno z lastnostmi, ki jih prikazuje slika 3.3.

S spustnim menijem Tip skale moramo določiti tip skale kriterija, kjer izbiramo med opcijama Fixed in Relative. Za uporabo je najenostavneje, če izberemo relativni tip skale. Ta poskrbi, da se dodeli vrednost 0 najslabši varianti in vrednost 100 najboljši. Relativni tip je torej dobro uporabiti, če za odločitev želimo porabiti čim manj časa. Pri fiksnem tipu skale pa robni točki določimo z vnosom vrednosti v polje Min in Max. V ti dve točki ponavadi vstavimo najboljšo in najslabšo vrednost, ki se za dan kriterij realistično še lahko pojavi. Fiksni tip zahteva več časa za določanje dobrih robnih točk,



Slika 3.3: Urejanje kriterija.

vendar lahko mejni točki določimo pred dobrim poznavanjem variant in se že pred vnosom vrednosti variant lahko lotimo uteževanja modela. V praksi se lahko zgodi, da smo nastavili vrednost tipa skale na fiksno in določili spodnjo ter zgornjo mejo, ki jo varianta, vnesena kasneje, ne upošteva - je izven intervala vrednosti. V splošnem lahko spremenimo mejni točki, kar vpliva na model. Lahko pa prekoračeno vrednost spremenimo v največjo dovoljeno, če je ta prevelika in na najmanjšo dovoljeno, če je ta premajhna. Tako se bo vrednost preslikala v koristnost kriterija 100 oz. 0. Aplikacija v takem primeru kršitve postavljenih mej opozori uporabnika z rdeče obarvanim poljem v preglednici variant.

Naslednji spustni meni služi za določanje tipa funkcije koristnosti kriterija in ponuja možnosti izbire: Linear, Piecewise, Discrete. To lastnost najlažje določimo glede na tip podatkov, ki jih kriterij predstavlja. V našem primeru

lahko opazimo, da je privlačnosti določena diskretna funkcija koristnosti. Privlačnost ne predstavlja nekih direktno merljivih podatkov in jo je težko v to tudi pretvoriti. Zato bomo vrednotenje privlačnosti kasneje naredili z metodo MACBETH, ki v naši metodi zahteva diskretno funkcijo koristnosti. Vrednosti privlačnosti so kar imena variant, da jih bomo lažje primerjali kasneje.

S potrditvenim stikalom Obratna skala, lahko obrnemo preslikavo vrednosti, kjer najvišja varianta dobi vrednost 0 in najnižja 100. Ponavadi je polje Obratna skala izbrano ob ceni in podobnih kriterijih, saj nam najdražja varianta predstavlja najslabšo možnost. Pojavno okno vsebuje še polje Opis, ki je zgolj informativne narave, če želimo h kriteriju zapisati poljuben komentar.

Za podani primer končne nastavitve lastnosti kriterijev prikazuje tabela 3.1(naj spomnim, da je kriterij podprti protokoli odstranjen):

Kriteriji	**Skala**	**Tip fun.**	**Min**	**Max**	**Inverzno**
Št. jeder	fiksna	odsekovna	2	8	NE
Hitrost	relativna	linearna	Min	Max	NE
RAM	fiksna	diskretna	0	100	NE
Velikost zas.	relativna	linearna	Min	Max	NE
Čas priprav.	relativna	linearna	Min	Max	NE
Privlačnost	fiksna	diskretna	0	100	NE
Cena	ralativna	linearna	Min	Max	DA

Tabela 3.1: Lastnosti kriterijev za podani primer.

3.5 Funkcije koristnosti

Kriterijem smo tipe funkcij že določili, zdaj pa je treba določiti še način preslikave iz vrednosti kriterija v koristnost kriterija. Vsi kriteriji, ki smo jim že pred tem določili linearni tip funkcije koristnosti, imajo funkcijo že dokončno definirano in ni treba narediti več ničesar. Za take kriterije aplikacija samo

informativno v pojavnem oknu prikaže linearni grafa s točkami, ki predstavljajo variante.

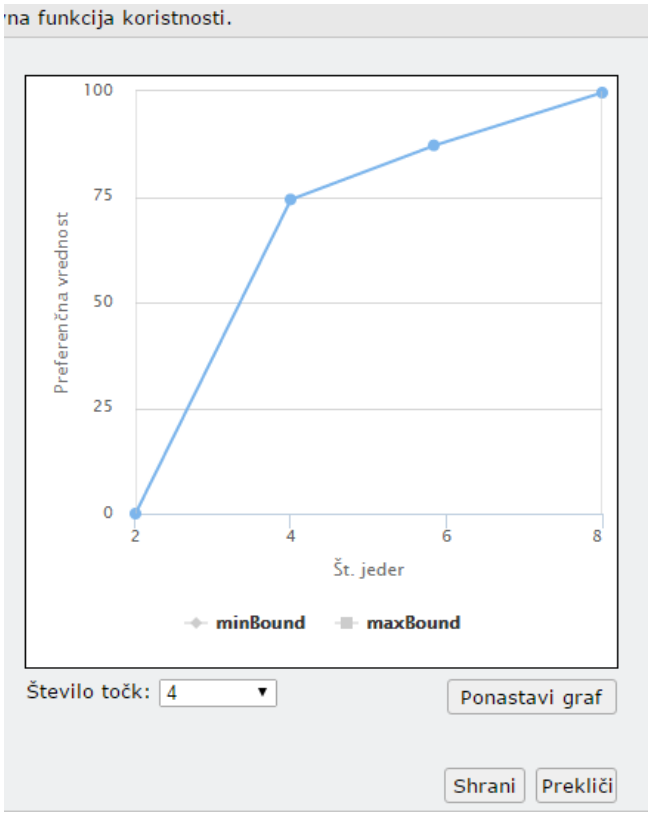
3.5.1 Odsekovna funkcija koristnosti

Preostane nam torej nastavitev kriterijev z odsekovnimi in diskretnimi funkcijami koristnosti. Slika 3.4 prikazuje nastavitev odsekovne funkcije kriteriju "Št. jeder". S slike je razvidno, da je najbolj zaželen telefon z osmimi jedri in najmanj z dvema. Ker smo ocenili, da naša delovna aplikacija na telefonu s štirimi jedri dela izrazito počasneje, kot aplikacija z štirimi jedri ali več, si bolj želimo telefon z vsaj štirimi jedri. Razlika od štirih do osmih jeder nam ne predstavlja velike vrednosti, saj se aplikacija na napravi s štirimi jedri izvaja dovolj tekoče.

3.5.2 Diskretna funkcija koristnosti

Čeprav kriterij RAM predstavlja številske podatke, ima določeno diskretno funkcijo koristnosti, saj imajo navadno naprave od 1 do 4 GB delovnega pomnilnika, kar lahko predstavimo s štirimi kategorijami: 1 GB, 2 GB, 3 GB, 4 GB. Slika 3.5 prikazuje nastavitev diskretne funkcije koristnosti kriteriju RAM. Sprva moramo vnesti kategorije, kar lahko storimo preko polja Ime kategorije in klika na gumb Dodaj kategorijo. Kadar pa imamo v aplikaciji že vnesene variante in njihove vrednosti, za diskretni kriterij, ki mu nastavljamo preslikavo, lahko z enim klikom na ikono  vnesemo vse obstoječe vrednosti kriterija iz trenutnega nabora variant.

Priporočljivo je, da sprva najbolj zaželeni kategoriji nastavimo vrednost 100, šele potem subjektivno presodimo in nastavimo ostale kategorije glede na prvotno izpolnjeno kategorijo. Če nobena izmed kategorij ne bo imela vrednosti 100, bo to vplivalo na model, kjer bo tako nastavljen kriterij predstavljal manjšo moč, česar bi se morali zavedati pri uteževanju. Ikoni  in  omogočata sortiranje nastavljenih kategorij po vrednosti padajoče in



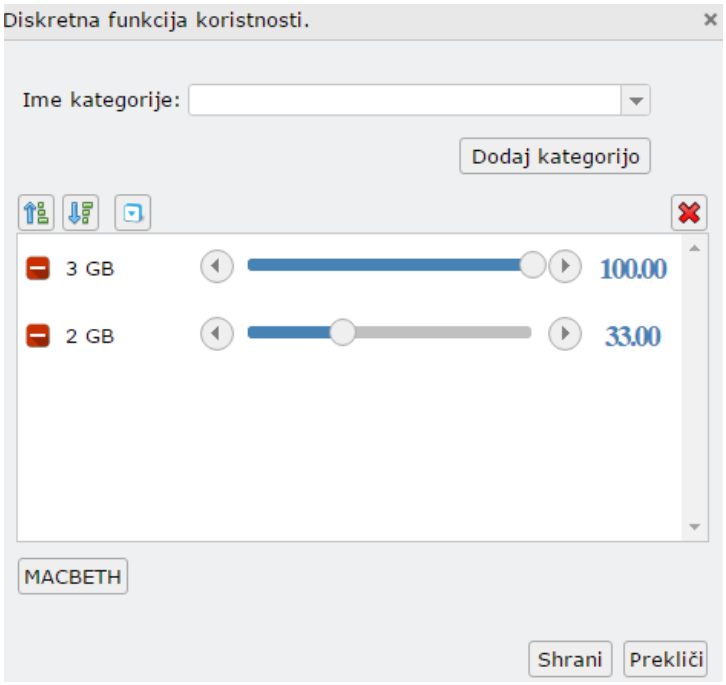
Slika 3.4: Prikaz nastavitve odsekovne funkcije.

naraščajoče.

V našem primeru imamo vstavljeni samo kategoriji 2 GB in 3 GB, kjer nam je naprava s 3 GB trikrat bolj pomembna kot naprav z 2 GB prostora delovnega pomnilnika.

3.5.3 Diskretna funkcija koristnosti z metodo MACBETH

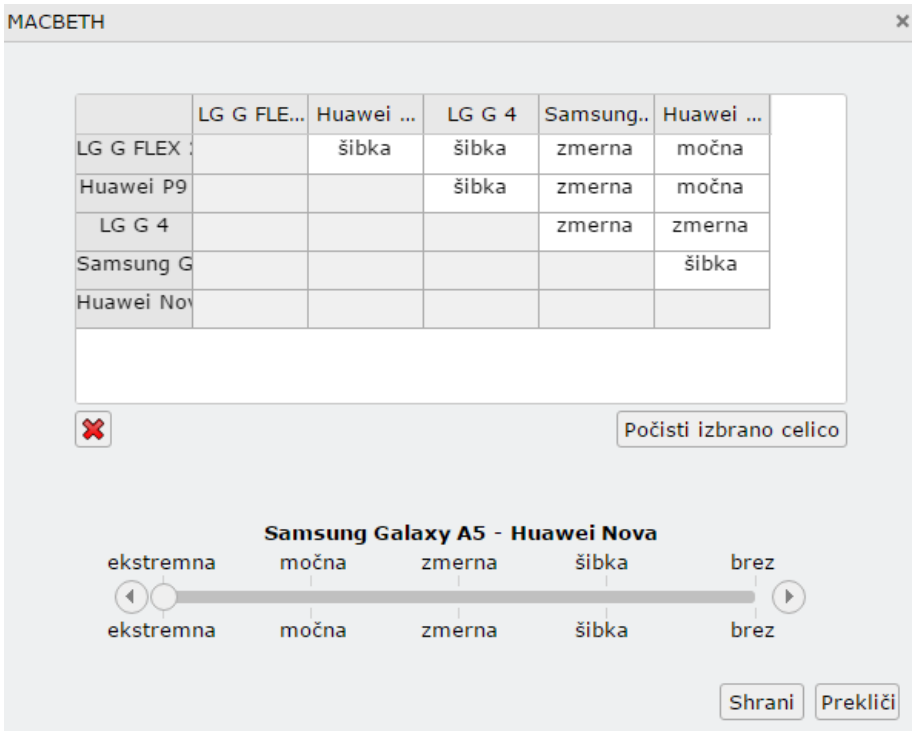
Kriteriju Privlačnost bomo nastavili vrednosti s pomočjo uporabe metode MACBETH, ki je iz teoretičnega vidika delovanja procesa podrobneje opisana v poglavju 5. Za uporabo MACBETH-a kliknemo na gumb MACBETH, ki je na pojavnem oknu diskretne funkcije koristnosti. Prikaže se nam novo okno, ki ga prikazuje slika 3.6.



Slika 3.5: Prikaz nastavitve diskretne funkcije.

Sprva je bilo treba razporediti telefone od najbolj do najmanj privlačnega. To najlažje storimo s klikom na naslov stolpca, nakar se nam prikaže pojavni spustni meni, kjer izberemo napravo, za katero želimo da je na mestu kliknjenega stolpca. Aplikacija omogoča menjavo stolpcev tudi z uporabo primi in potegni funkcionalnosti. Pomembno je, da so v tabeli elementi razporejeni po preferenci od leve proti desni padajoče.

Ko imamo po privlačnosti razporejene telefone, moramo mobilne naprave paroma primerjati po subjektivni moči razlike privlačnosti in jim temu primerno določiti eno izmed kategorij: ekstremna, močna, zmerna, šibka, brez. To storimo s klikom na celico, ki povezuje elementa, ki ju želimo primerjati. Ko se pod tabelo izpišeta imeni elementov, ki ju primerjamo, z drsnikom določimo kategorijo razlike, ki se izpiše tudi v izbrani celici tabele. Po končanem primerjanju delo shranimo s klikom na gumb OK. Rezultati se prikažejo na drsnikih pojavnega okna diskretne funkcije koristnosti. Po



Slika 3.6: Prikaz uporabe metode MACBETH.

želji lahko vrednosti spreminjamo znotraj dovoljenih intervalov, ki ne kršijo omejitev, postavljenih v MACBETH-tabeli.

V našem primeru je najbolj privlačen mobilnik LG G FLEX 2 najmanj privlačen pa Huawei Nova. S slike je razvidno, da je razlika med najbolj in najmanj privlačnim telefonom ocenjena kot močna (v zgornjem desnem kotu). Koristnosti privlačnosti telefonov so za naš primer številsko ovrednotene:

- LG G FLEX 2 = 100 (na intervalu: 100-100)
- Huawei P9 = 85.71 (na intervalu: 71,44-99,99)
- LG G 4 = 71.43 (na intervalu: 64,29-85,00)
- Samsung Galaxy = 28.57 (na intervalu: 14,30-35,71)

- LG G FLEX 2 = 0 (na intervalu: 0-0)

3.5.4 Nastavljanje uteži

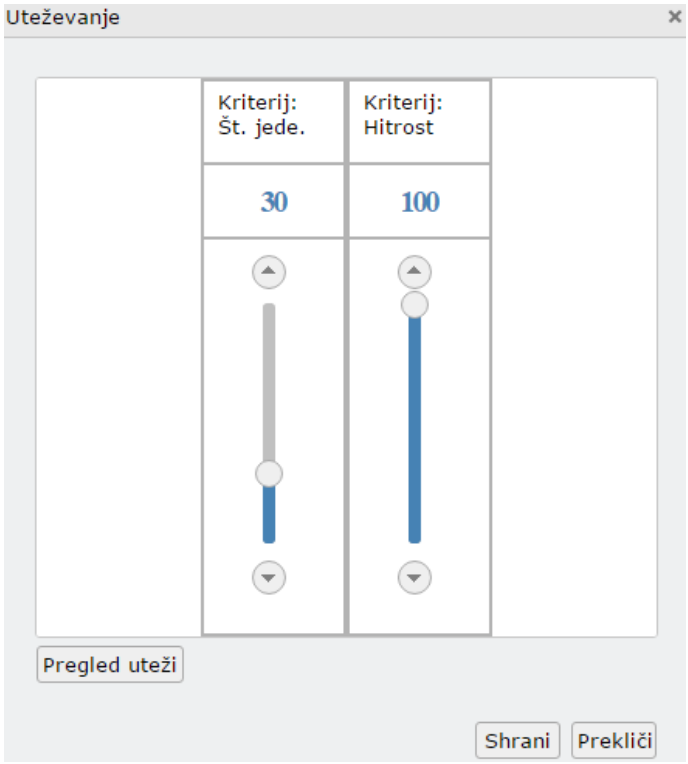
Sledi uteževanje odločitvenega drevesa. Sprva so bili uteženi kriteriji na najnižjem nivoju. Pri družini CPU tako ocenjujemo kriterija št. jeder in hitrost. Hitrost ima razpon kar od 1,5 GHz do 2,5 GHz, kar ni zanemarljivo in predstavlja zelo veliko razliko v zmogljivosti procesorja. Skala števila jeder se začne z dvema in konča z osmimi jedri, kar tudi ni majhna razlika. Vendar so bila jedra nastavljena kot kriterij s fiksno skalo (bolj zaradi namena primera) in v resnici naše variante tvorijo razpon od šest do osem jeder, kar je bistveno manj. Menimo, da je pri danem razponu vrednost 1 GHz precej bolj pomembna kot vrednost dveh jeder. Tako ima družina CPU nastavljen kot najbolj pomemben kriterij hitrost in je ovrednoten z vrednostjo 100. Številu jeder smo določili utež 30.

Nastavitev uteži aplikacija ponuja v pojavnem oknu, ki je prikazano na sliki 3.7. Vnos uteži je možen preko drsnikov ali z direktnim vnosom števila.

Potem smo se premaknili na ocenjevanje družine Zmogljivost. Oceniti je potrebno med kriterijema RAM in CPU. Ker je CPU vozlišče, vzamemo za primerjavo njegovega najpomembnejšega potomca, ki je v našem primeru kriterij Hitrost. Postopek ponavljamo, dokler niso uteženi vsi kriteriji.

Aplikacija nam v pojavnem oknu za nastavitev uteži ponuja zgolj informacije o nastavljenih relativnih utežeh (uteži ene družine na enem nivoju), kar je v primeru večjega števila uteži premalo, saj je pravilna interpretacija uteževanja zelo pomembna. S tem namenom se ob kliku na gumb Pregled uteži prikaže preglednica, ki je prikazana na sliki 3.8. Okno prikazuje tako relativne kot končne normalizirane uteži kriterijev, ki so v listih. Poleg tega prikazuje tudi razpon in tip skale. Tako nam je uteževanje z dostopom do teh informacij olajšano. Pregled uteži je koristen predvsem na koncu uteževanja, kadar preverjamo, ali smo bili pri nastavljanju dosledni.

Slika 3.8 prikazuje stanje po koncu uteževanja našega modela. Iz preglednice je tako možno razbrati, da je med vsemi kriteriji najpomembnejši



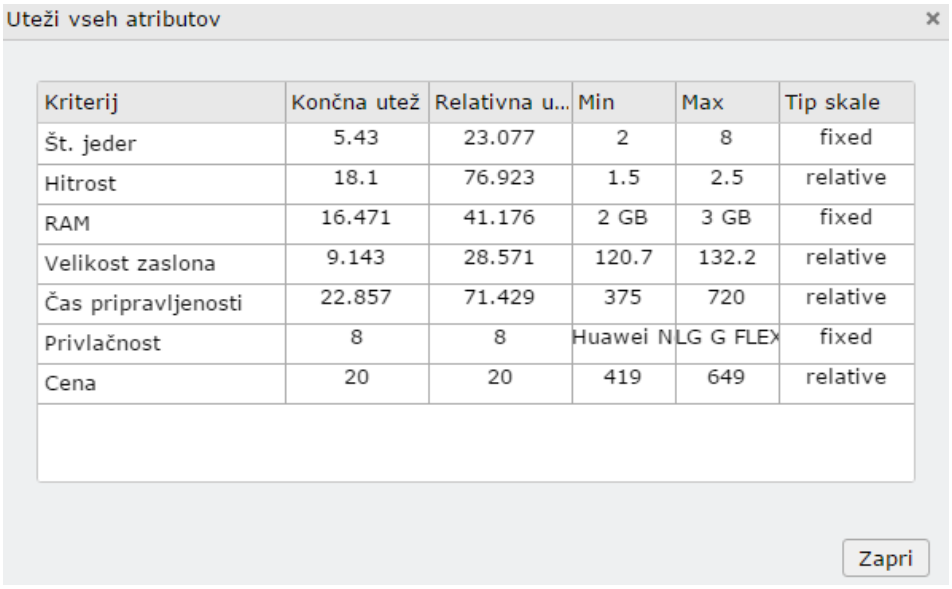
Slika 3.7: Prikaz nastavitve uteži.

kriterij Čas pripravljenosti z normalizirano utežjo 22,857 in najmanj pomemben Privlačnost z utežjo 8. Uporabniško nastavljene uteži so: Hitrost=100, Št. jeder=30, CPU=100, RAM=70, Velikost zasl.=40, Čas prip.=100, Zmogljivost=100, Uporabnost=80, Privlačnost=20 in Cena=50.

3.5.5 Izračun koristnosti

Iz do zdaj vnesenih podatkov je možno izračunati koristnosti variant. Aplikacija pri analizi izračuna koristnosti variant z modelom utežene vsote 2.1.

Primer izračuna koristnosti za varianto Huawei P9 je podan spodaj. Vrednosti Huawei P9 se preslikajo v delne koristnosti kriterijev: Št. jeder = 100, Hitrost = 100, RAM = 100, Velikost = 82, 61, Čas pripr. = 55, 36,



Slika 3.8: Preglednica uteži osnovnih kriterijev.

Privlačnost $= 85,71$, Cena $= 39,13$

$$Koristnost(HuaweiP9) = 100 * 0,054 + 100 * 0,181 + 100 * 0,165 + 82,61 * 0,091 + 55,36 * 0,229 + 85,71 * 0,08 + 39,13 * 0,20 = 74,88 \quad (3.1)$$

Koristnosti ostalih variant podanega primera pridobljene po enakem postopku:

- Huawei P9: 74,88
- Huawei Nova: 73,86
- Samsung Galaxy A5: 60,33
- LG G 4: 42,24
- LG G FLEX 2: 31,83

Telefon Huawei P9 si je pridobil najvišjo koristnost. Odločitvenega procesa še ni konec, zato se še ne moremo odločiti, da bomo najbolje ocenjeno

napravo tudi zares kupili. Za postopke, ki nas privedejo do končne odločitve, lahko sežete po diplomskem delu [2].

Poglavje 4

Razvoj aplikacije

Glavni namen razvoja aplikacije je bil izdelava programskega orodja za podporo odločitvenemu procesu pri učenju metode večparametrskega odločanja pri predmetu Odločitveni sistemi. Temu primerno so tudi implementirane potrebne funkcionalnosti. Pri naboru izdelanih funkcionalnosti, je bil v pomoč tudi sicer komercialni program Hivew3 [12], ki se za namen poučevanja uporablja pri vajah omenjenega predmeta.

Razvita aplikacija je odprtokodna in dostopna na spletni strani GitHub, preko katere je izdelek možno prenesti in ga po potrebi tudi nadgraditi. Skrbnik repozitorija je as. dr. Martin Možina, ki vodi vaje omenjenega predmeta in je kot somentor sodeloval pri izdelavi aplikacije. Povezava do repozitorija: `https://github.com/martinmozina/friview`.

Poglavje predstavi izbor orodij in nekaj razvojnih odločitev, ki so se pojavile med samim razvojem.

4.1 Izbor jeziokv Javascript, HTML in CSS

Pri izboru tehnologij nas je usmerjala misel na aplikacijo, ki jo bo moč uporabljati na različnih sistemih. S tem namenom je bil izbran skriptni programski jezi JavaScript [13] v kombinaciji z označevalnim jezikom HTML [15] (Hyper Text Markup Language) in oblikovnim jezikom CSS [14](Cascading Style

Sheets). Tako je aplikacijo možno uporabljati na sistemih, ki podpirajo spletne brskalnike. Ker aplikacija uporablja tehnologije, ki se poganjajo samo na odjemalčevi strani, za delovanje aplikacije strežnik in dostop do interneta nista nujno potrebna. Za uporabo aplikacije je najbolj priporočljiv brskalnik Chrome, saj sem ga sam pri izdelavi uporabljal v največji meri. Funkcionalnosti aplikacije so bile testirane tudi na brskalniku Firefox.

Zaradi varnostnih razlogov imajo spletni brskalniki zelo omejen dostop do datotečnega sistema na napravah, kjer so nameščeni. Zaradi tega JavaScript teče v nadzorovanem okolju oz. peskovniku [11]. To se izkaže kot slabost, kadar izdelujemo aplikacijo, ki potrebuje pravice za zapis datotek, kar pomeni, da aplikacija ne more shranjevati datotek z direktnim dostopom do datotečnega sistema. Za obhod problema sta bila izbrana vmesnika [16] FileSaver.js in Blob.js. Oba omenjena vmesnika implementirata enakoimenska W3C programska vmesnika in sta podprta na vseh modernih brskalnikih. Objekt Blob [18] predstavlja nespremenljive datotečne podatke, ki jih je možno shraniti z uporabo FileSaverjeve [17] funkcije saveAs(blobObject, filename). Funkciji se poda Blob objekt in ime datoteke in ne njena pot. Datoteka se prenese v direktorij, ki ga ima brskalnik nastavljenega za prenose.

Slabost takega načina shranjevanja datoteke je, da se ob vsaki shrambi naredi nova datoteka in imamo tako v primeru večkratnega shranjevanja več datotek in ne le ene, ki bi jo želeli ob shranitvi prepisati oz. dopolniti.

V veliko pomoč so bile prosto dostopne knjižnice za uporabo:

- **jQWidgets**[19]: izdelava večine gradnikov (okna, drsniki, tabele ...),
- **D3.js**[21]: knjižnica sicer namenjena vizualizaciji podatkov je bila uporabljena za izris odločitvenega drevesa,
- **Highcharts**[20]: izdelava interaktivnih grafov pri funkcijah koristnosti,
- **jsLPSolver**[23]: reševanje linearnih programov,
- **algebra.js**[24]: okrajšava in izračun matematičnih izrazov,
- **SheetJS, xls, xslx**[22]: rokovanje s tabelarnimi datotekami.

4.2 Funkcionalnost razveljavi/ponovno uveljavi

Pri delu s programom je koristna možnost vračanja na prejšnja stanja in obnavljanja že opravljenih korakov. Za uporabnika bi bilo zelo neprijazno, če bi moral za vsako napako brisati in ponovno graditi model, ki je že bil zgrajen. Tako početje bi nam vzelo kar precej časa po nepotrebnem. Zato je funkcionalnost razveljavi/ponovno uveljavi implementirana tudi pri gradnji modela.

Za implementacijo te funkcionalnosti sta v splošnem koristna dva različna načrtovalska vzorca: ukazni vzorec (command pattern) in spominski vzorec (memento pattern). Odločiti se je bilo treba za enega izmed njiju. Sledi opis obeh z njunimi prednostmi in slabostmi ter končna odločitev izbire.

4.2.1 Ukazni vzorec

Z ukaznim vzorcem [25] pridobimo način, da inkapsuliramo klice akcij (metod, funkcij) v objekte, ki se hranijo in jih je mogoče ponovno izvršiti v kasnejšem času. Vzorec vsebuje pet elementov [26]:

- Ukaz (Command): določa vmesnik za izvršitev operacije.
- KonkretenUkaz (ConcreteCommand): implementira vmesnik Ukaz.
- Prejemnik (Receiver): izvrši operacijo, ki je podana s strani odjemalca.
- Odjemalec (Client): kreira ukaze in jih povezuje s prejemniki.
- Klicatelj (Invoker): od ukaza zahteva izvršitev zahteve.

Za implementacijo razveljavi/ponovno uveljavi je treba za vsako akcijo narediti obratno akcijo, ki bi se izvršila ob razveljavitvi ukaza. Slabost ukaznega vzorca je večje število majhnih razredov, ki hranijo sezname ukazov.

4.2.2 Spominski vzorec

Vzorec omogoča hranjenje stanja objektov. Sestavljen je iz treh elementov:

- Tvorec (Originator): je objekt z notranjim stanjem, ki ga želimo shraniti in obnoviti.

- Spomin (Memento): je objekt, ki ga je možno shraniti in predstavlja shranjeno stanje tvorca.

- Skrbnik (Caretaker): je odgovoren za shranjevanje in obnavljanje objektov Memento.

Razveljavi/ponovno uveljavi je možno implementirati tako, da pred vsako akcijo, ki spremeni model, shranimo celotno stanje modela. Kasneje ob razveljavitvi ukaza obnovimo stanje modela s prepisom trenutnega stanja s shranjenim stanjem. Če je naš objekt, katerega stanja želimo obnoviti, prostorsko zahteven (npr. slika) moramo pri uporabi spominskega vzorca ob vsaki spremembi shraniti stanje, ki zavzame veliko prostora. Pri podpori funkcionalnosti razveljavi/ponovno uveljavi pa je zaželeno omogočeno shranjevanje kar precej stanj, saj želimo omogočiti vračanje za precejšnje število korakov. Tako potreba po prostoru začne hitro naraščati. Torej je ta načrtovalski vzorec v primerih s prostorsko zahtevnimi stanji popolnoma neprimeren. Njegova velika prednost glede na ukazni vzorec pa je dokaj enostavna implementacija in uporaba.

Pri izdelavi aplikacije je bil izbran spominski vzorec, saj je v našem primeru tvorec odločitveni model in spominski objekt predstavlja niz v obliki JSON. Tako eno stanje zavzame zgolj nekaj kilobajtov. Torej imamo na razpolago veliko prostora za hranjenje večjega števila stanj. Zato je pri gradnji modela funkcionalnost razveljavi/ponovno uveljavi realizirana z različico spominskega vzorca.

4.2.3 Uvoz variant

V praksi si včasih želimo nabor variant sprva vnesti in urejati s programom Excel. Vnos variant v Excelu pride prav, kadar si želimo hraniti variante v ločeni datoteki ali pa se nam urejanje variant zdi bolj preprosto s programom Excel. Razlog za to je tudi lahko, ker trenutno nimamo dostopa do aplikacije za odločanje. Iz teh razlogov je v okviru te diplomske naloge in gradnje modela implementirana tudi funkcionalnost uvoza variant iz datotek tipa .xls in .xlsx.

Opis oblike datoteke, ki jo uvažamo, je opisan ob primeru v poglavju 3.3.

Poglavje 5

MACBETH

Metoda MACBETH je lahko pri odločitvenih problemih v splošnem uporabljena za različne namene. Moč jo je koristiti kot samostojno odločitveno metodo s samostojnim procesom odločanja. V splošnem se uporablja tudi za izpeljavo uteži pri odločitvenem drevesu.

Kadar pri odločitvenem procesu naletimo na problem, ko ne moremo vrednosti kriterijev s pomočjo funkcije koristnosti direktno preslikati v koristnost, je uporaba metode MACBETH močno dobrodošla. Z njeno pomočjo lahko vrednostim takih kriterijev določimo koristnosti na podlagi subjektivnih presoj na konsistenten način, kar bi bilo brez uporabe MACBETH-algoritma dokaj zahtevna operacija. Iz teh razlogov je algoritem MACBETH implementiran tudi v izdelani aplikaciji.

Definicija [5]
MACBETH je interaktivni pristop, ki vodi konstrukcijo, na množici elementov X, intervalne merske lestvice, ki kvantificira privlačnost elementov množice X na podlagi odločevalčevega mnenja.

5.1 MACBETH-postopek

Naj bo X končna množica elementov (variant, kategorij), ki jih želimo primerjati iz vidika njihove privlačnosti oz. zaželenosti. Množica vsebuje vsaj dva elementa.

V prvem koraku moramo elemente množice X razporediti po privlačnosti od najbolj privlačnega do najmanj privlačnega. Elemente lahko tako uredimo direktno, pri večjem številu elementov pa si lahko pomagamo z zaporedjem vprašanj [3]: "Ali je kateri od dveh elementov bolj privlačen kot drugi?" Če si odgovorimo z da, nadaljujemo z vprašanjem: "Kateri izmed elementov je bolj privlačen?" Na tak način pridobimo ordinalne informacije o želenosti elementov. Vendar nam ne zadošča samo znanje, kateri element je boljši od drugega, za pravilno skalo moramo vedeti tudi, za koliko je boljši.

Ker nam je težko določiti intervalno mersko lestvico direktno (in pri tem ostati konsistenten) nam MACBETH tudi pri drugem koraku olajša delo s postavitvijo več enostavnih vprašanj. Za vsak par x in y iz množice X, kjer je x bolj privlačen kot y, se sprašujemo o razliki privlačnosti med elementoma. Odgovoriti si moramo na vprašanje: "Koliko močna je za nas razlika vrednosti med elementom x in elementom y?" Kot odgovor pa imamo tradicionalno na razpolago naslednjih sedem možnosti:

- C_0 **brez** razlike v privlačnosti,
- C_1 **zelo šibka** razlika v privlačnosti,
- C_2 **šibka** razlika v privlačnosti,
- C_3 **zmerna** razlika v privlačnosti,
- C_4 **močna** razlika v privlačnosti,
- C_5 **zelo močna** razlika v privlačnosti,
- C_6 **ekstremna** razlika v privlačnosti.

Včasih se med iskanjem odgovora težko odločimo, katero kategorijo razlike bi pripisali dvema elementoma (npr. težko nam je preceniti, ali je razlika šibka ali samo zelo šibka). Zato sta v aplikaciji odstranjeni izbiri zelo šibka in zelo močna razlika. Tako preidemo na samo pet možnih odgovorov in na tak način odločevalcu olajšamo izbor odgovora ter poenostavimo postopek ocenjevanja razlik privlačnosti.

Pri MACBETH-postopku se za vnos podatkov ponavadi uporablja tabela (oz. matrika) presoj, ki omogoča dovolj jasen pregled in enostaven vnos podatkov. Stolpci in vrstice predstavljajo posamezne elemente množice X. Stolpce moramo razporediti po privlačnosti, kjer je prvi element z leve strani najbolj privlačen in skrajno desni element najmanj privlačen. Elementi v vrsticah so razporejeni enako kot elementi v stolpcih z najbolj privlačnim elementom na vrhu. Primer prikazuje izpolnjena tabela 5.1, kjer je množica elementov $X = \{A, B, C, D\}$, ki imajo naslednje zaporedje privlačnosti: $A > B > C > D$.

	A	**B**	**C**	**D**
A	brez	šibka	zmerna	močna
B		brez	šibka	zmerna
C			brez	zmerna
D				brez

Tabela 5.1: Primer tabele razlik privlačnosti elementov.

Na diagonali tabele bo vedno vnesena vrednost **brez**, saj se element glede na samega sebe ne more razlikovati. Spodnja leva polovica tabele je običajno neizpolnjena, ker bi sicer vsebovala enake vrednosti kot zgornja desna polovica in bi s tem zmanjšali preglednost nad tabelo. Taka tabela prikazuje vse potrebne informacije, ki jih moramo vnesti za preračun vrednosti elementov.

Ko MACBETH-metoda konča z izračuni, nadaljujemo postopek na tretjem koraku, opisanem v podpoglavju 5.3.

5.2 Izračun MACBETH-lestvice

V tem delu je opisan računski postopek, ki pretvori matriko razlik privlačnosti v numerično MACBETH-lestvico μ.

V prvem koraku postopka, ko razporedimo elemente po privlačnosti, pridobimo ordinalne informacije, kar pomeni, da je možno vsakemu elementu množice X dodeliti vrednost $\mu(x)$, ki zadovolji ordinalne merilne pogoje:

Pogoj 1 (ordinalni pogoj):

- $\forall x, y \in X$: [$\mu(x) > \mu(y) \Leftrightarrow x$ je bolj privlačen kot y].
- $\forall x, y \in X$: [$\mu(x) = \mu(y) \Leftrightarrow x$ in y sta enako privlačna].

Po drugem koraku, ko smo konsistentno določili razlike privlačnosti med elementi množice X (in iz prvega koraka razpored po privlačnosti), je možno vsakemu elementu izmed množice X določiti vrednost $\mu(x)$, ki zadovolji kardinalne merilne pogoje:

Pogoj 2 (semantični pogoj):

- $\forall x, y \in X$: [$\mu(x) > \mu(y) \Leftrightarrow x$ je bolj privlačen kot y].
- $\forall x, y \in X$: [$\mu(x) = \mu(y) \Leftrightarrow x$ in y sta enako privlačna].
- $\forall k, k' \in \{1, 2, 3, 4, 5, 6\}$, $\forall x, y, w, z \in X$, kjer $(x, y) \in C_k$ in $(w, z) \in C_{k'}$ velja:
$$k \geq k' + 1 \Rightarrow \mu(x) - \mu(y) > \mu(w) - \mu(z)$$

Če se **pogoja 1** in **pogoja 2** ne da zagotoviti, pomeni, da smo bili ob vnašanju podatkov nekonsistentni in moramo vnos popraviti. Kadar taka skala obstaja, jo MACBETH določi z rešitvijo linearnega programa. V aplikaciji se uporablja podan linearni program [4]:

Množica elementov: $S = \{x_1, x_2, ..., x_n\}$

Pozitivne spremenljivke: $\mu(x_i), i \in \{1, 2, ...n\}$

Namenska oz. ciljna funkcija: $Min(\mu(x_1))$

Omejitve:

- omejitev 1: $\mu(x_n) = 0$,
- omejitev 2: $\forall x, y \in X$: $xIy \Rightarrow \mu(x) = \mu(y)$,
- omejitev 3: $\forall x, y \in X$: $xPy \Rightarrow \mu(x) \geq \mu(y) + C_{x,y}$,
- omejitev 4: $\forall k, k' \in \{1, 2, 3, 4, 5, 6\}$, kjer $k > k'$,
  $\forall (x, y) \in C_k$ in $\forall (w, z) \in C_{k'}$:
  $\mu(x) - \mu(y) \geq \mu(w) - \mu(z) + (C_{x,y} - C_{w,z})$

Definicije:

- P je asimetrična in negativno tranzitivna relacija nad množico X, ki modelira razpored elementov množice X padajoče po privlačnosti.
- I je binarna relacija nad množico X, kjer $\forall x, y \in X : xIy \Leftrightarrow x\neg Py$ in $y\neg Px$.
- $C_{x,y}$ je številska razlika v privlačnosti med elementoma x in y.

Aplikacija tako za primer podatkov iz tabele 5.2 sestavi podane omejitve, ki so uporabljene za reševanje linearnega programa:

$\mu(A) \geq \mu(D) + 4;$
$\mu(A) \geq \mu(C) + 3;$
$\mu(B) \geq \mu(D) + 3;$
$\mu(C) \geq \mu(D) + 3;$
$\mu(A) \geq \mu(B) + 2;$
$\mu(B) \geq \mu(C) + 2;$

$\mu(A) - \mu(D) \geq \mu(A) - \mu(C) + 4 - 3;$
$\mu(A) - \mu(D) \geq \mu(B) - \mu(D) + 4 - 3;$
$\mu(A) - \mu(D) \geq \mu(C) - \mu(D) + 4 - 3;$
$\mu(A) - \mu(D) \geq \mu(A) - \mu(B) + 4 - 2;$
$\mu(A) - \mu(D) \geq \mu(B) - \mu(C) + 4 - 2;$
$\mu(A) - \mu(C) \geq \mu(A) - \mu(B) + 3 - 2;$
$\mu(A) - \mu(C) \geq \mu(B) - \mu(C) + 3 - 2;$
$\mu(B) - \mu(D) \geq \mu(A) - \mu(B) + 3 - 2;$
$\mu(B) - \mu(D) \geq \mu(B) - \mu(C) + 3 - 2;$
$\mu(C) - \mu(D) \geq \mu(A) - \mu(B) + 3 - 2;$
$\mu(C) - \mu(D) \geq \mu(B) - \mu(C) + 3 - 2;$

Tabela 5.2 prikazuje relativne razlike iz table 5.1, preslikane v številske vrednosti kategorij moči razlik, ki so uporabljene kot vhod v generiranje omenjenih omejitev linearnega programa.

	A	**B**	**C**	**D**
A	0	2	3	4
B		0	2	3
C			0	3
D				0

Tabela 5.2: Številske vrednosti kategorij razlik med pari primerjajočih elementov.

Linearni program vrne MACBETH-skalo μ, ki predstavlja ovrednotene elemente množice X. Primer ovrednotenih elementov množice X za podatke iz tabele 5.2:

- $\mu(A) = 7,$
- $\mu(B) = 5,$
- $\mu(C) = 3,$

	A	**B**	**C**	**D**
A	0	2	4	7
B		0	2	5
C			0	3
D				0

Tabela 5.3: Številske vrednosti končnih razlik med pari primerjajočih elementov.

- $\mu(D) = 0$.

Tabela 5.3 predstavlja razlike med vrednostmi elementov, pridobljenih z rešitvijo linearnega programa. S pomočjo tabele 5.3 in 5.2 lahko opazimo, da se kategorije razlik, ki smo jih dodelili, ne preslikajo enolično v vrednost končne razlike. Kot odločevalci smo parom (A, C), (B, D) in (C, D) določili kategorijo razlike **zmerno** oz. C_3, vendar se ta kategorija preslika v različne vrednosti razlik:

- $\mu(A) - \mu(C) = 4$,
- $\mu(B) - \mu(D) = 5$,
- $\mu(C) - \mu(D) = 3$.

Tako je, ker MACBETH-skala kategorijam moči razlik C_i dodeli intervale [5]. Minimizacija namenske oz. ciljne funkcije linearnega programa poskuša skrčiti intervale in razlike med njimi, kolikor je to mogoče. Zaradi omejitve 4 pa je razdalja med intervali lahko najmanj 1.

5.3 Prehod na intervalno mersko lestvico

MACBETH-lestvico, pridobljeno v 5.2, moramo zdaj obdelati še z vidika razmerij razlik med vrednostmi elementov množice X. Pri uporabi MACBETH-metode za namen določanja sestavljene skale kriterijev variant to predstavlja tudi zadnji korak, ki ga moramo opraviti.

Program sprva spremeni MACBETH-skalo v transformirano MACBETH-skalo ν z uporabo linearne interpolacije. Tako so nam pridobljene vrednosti in razmerja med elementi množice lažje predstavljivi. Pri pretvorbi program določi najslabšemu elementu množice X vrednost 0, najboljšemu pa vrednost 100. Tako bi se vrednosti iz našega primera preslikale v:

- $\mu(A) = 7 \Rightarrow \nu(A) = 100$,
- $\mu(B) = 5 \Rightarrow \nu(B) = (5/7) * 100 = 71$,
- $\mu(C) = 3 \Rightarrow \nu(C) = (3/7) * 100 = 42$,
- $\mu(D) = 0 \Rightarrow \nu(D) = 0$.

MACBETH-izračun nam poda vrednosti posameznih elementov, ki so izračunani na podlagi omejitev pogoja 1 in 2, ki izhajajo iz vnesenih podatkov v MACBETH-tabeli. Vendar ni nujno, da relativne razlike med vrednostmi elementov odražajo relativne razlike privlačnosti, ki jo imamo kot odločevalci v mislih. Tako MACBETH-postopek omogoča končno razpravo o razmerjih razlik med elementi.

Za četverico elementov x, y, w, z iz množice X, kjer xPy in wPz lahko izračunamo razmerje [5]:

$$\frac{\nu(x) - \nu(y)}{\nu(w) - \nu(z)}$$

Tako se moramo vprašati, ali razmerja odražajo našo predstavo o njih. Recimo, da opazujemo spodnje razmerje med elementi A, B in C:

$$\frac{\nu(A) - \nu(C)}{\nu(B) - \nu(C)} = \frac{100 - 42}{71 - 42} = \frac{58}{29} = 2$$

Vprašamo se, ali nam razlika med elementoma A in C res pomeni dvakrat toliko kot razlika med elementoma B in C. Če je naš odgovor ne, nam aplikacija ponuja možnost ročne nastavitve vrednosti določenega elementa. Vrednost posameznih elementov lahko premikamo samo znotraj intervalov. Intervali skrbijo, da pri premiku vrednosti ne pride do kršitve ordinalnega pogoja 1 in semantičnega pogoja 2 glede na podatke, ki smo jih vnesli v

prvih dveh korakih MACBETH-postopka. Pri uporabi aplikacije običajno ne računamo vseh razmerij, ampak si lahko pomagamo z vizualno predstavo razmerij med vrednostmi in tako vsi izračuni razmerij niso potrebni.

5.4 Izračun dovoljenih intervalov

V našem primeru dobi element B spodnjo mejo na podlagi vnesene šibke razlike med elementoma A in B ter zmerne razlike med elementoma C in D. Zaradi pogoja 2 mora držati:

$$\mu(C) - \mu(D) > \mu(A) - \mu(B)$$

To pomeni, da mora B v primeru: $\mu(C) = 3$, $\mu(D) = 0$ in $\mu(A) = 7$, imeti vrednost $\mu(B) > 4$. Torej interpolirana vrednost $57,14$ predstavlja spodnjo mejo elementa B.

Aplikacija v resnici izračuna možne vrednosti iz vseh intervalov z elementom B in pri spodnji meji vzame tisto, ki je najbližja vrednosti B ter ni večja od njegove vrednosti. Za zgornjo mejo intervala pa vzame tisto, ki je najbližja vrednosti B in ni manjša od njegove vrednosti.

5.5 Preverjanje nekonsistentnosti

Pri vnosu se nam lahko hitro zgodi, da v MACBETH-tabeli dodelimo razlike privlačnosti med elementi na tak način, da za generirane omejitve ni mogoče pridobiti rezultatov linearnega programa. To pomeni, da smo bili pri vnosu razlik nekonsistentni. Recimo, da imamo elemente x, y in z, kjer so xPy, xPz in yPz. Če ocenimo, da je razlika med x in y močna ter med x in z šibka, imamo nekonsistenten vnos. Oddaljenost med elementoma x in z ne more biti manjša od oddaljenosti med elementoma x in y, saj smo razporedili y kot privlačnejši element od elementa z.

Aplikacija nas bo v primeru nekonsistentnega vnosa opozorila, da je prišlo do napake. Od nas zahteva, da nekonsistentnost odpravimo pred nadaljnjim

vnašanjem razlik. V pomoč prikaže možne spremembe, s katerimi lahko dosežemo konsistenten vnos.

Program pridobi predloge sprememb na podlagi poskušanja. Ko zazna nekonsistentnost, začne preiskovanje sprememb vseh celic. Iterativno vsaki celici spreminja vrednost po eno stopnjo moči razlike navzgor oz. navzdol. Ko je najdena prva vrednost celice, ki popravi nekonsistentnost, program prekine preiskovanje celice. Najdene možne spremembe tako poda uporabniku z obarvanjem celic MACBETH-tabele in s prikazom najbližje vrednosti, ki odpravi nekonsistentnost.

Poglavje 6

Sklepne ugotovitve

Izdelek diplomskega dela ponuja pomoč pri izvedbi kakovostnega odločitvenega procesa z metodo večparametrskega odločitvenega modela. Diplomsko delo ponuja tudi pregled nad večparametrskim odločitvenim procesom, ki ga je treba dobro poznati za pravilno uporabo izdelanega projekta. Delo je osredotočeno na gradnjo odločitvenega modela in s primerom podrobneje prikaže postopek gradnje z uporabo izdelane aplikacije. V prvi vrsti je izdelek namenjen v izobraževalne namene, vendar ga je možno koristno uporabljati tudi sicer. Kot dodaten produkt vsebuje implementacijo ter podrobnejši opis metode MACBETH in opis njene izvedbe.

Ob izdelavi diplomske naloge sem se podrobneje spoznal s posameznimi koraki odločitvenega procesa in tako pridobil znanje, ki je potrebno za izpeljavo dobrega odločitvenega procesa, ki vodi do kakovostne odločitve. Menim, da mi bo pridobljeno znanje v življenju prišlo prav, saj bom ob zahtevnejših odločitvah zagotovo segel po izdelani aplikaciji.

Pridobil sem tudi nekaj izkušenj z delom na aplikacij, ki je v celoti nisem razvijal sam. Pri izdelavi je bilo treba opraviti kar nekaj sestankov s prof. Aleksandrom Sadikovim, as. dr. Martinom Možino in kolegom študentom Alešem Bokalom, na katerih smo usklajevali potrebe ter zahteve. Do začetka izdelave izdelka sem programski jezik Javascript poznal le bežno in sem ga tako imel priložnost bolje spoznati ob delu na projektu malo širšega obsega.

Izdelana aplikacija je odprtokodna in prostodostopna. Tako so po potrebi v prihodnosti omogočene razširitve. Prostora za možne razširitve je še veliko. Ena izmed želenih dopolnitev bi bila funkcionalnost generiranja poročil, ki bi po koncu odločitvenega postopka naredila poročilo z vsebino odločitvenega modela in ugotovitvami odločitvene analize. Tako bi pri predmetu, ki je namenjen učenju opisanih metod, študenti lažje naredili končno poročilo, asistenti pa bi za pregledovanje dobili poročila v standardnem formatu.

Zelo dobrodošla bi bila razširitev, ki bi aplikacijo bolj prilagodila tabličnim napravam. Sicer jo je na tablicah mogoče uporabljati, vendar pri tem uporabnik ne bo imel najboljše uporabniške izkušnje. Pri izdelavi gradnikov aplikacije je bilo izbrano orodje jQWidgets, ki omogoča razširitev spletne aplikacije na način odzivne zasnove spletne strani (responsive web design). S to razširitvijo bi lahko prilagodili videz gradnikov, ki so tabličnim napravam bolj domači, in na tak način dosegli izboljšano uporabniško izkušnjo.

Literatura

[1] A. Ishizaka, P. Nemery "Multi-Criteria Decision Analysis Methods and Software", str. 114–118, 2013.

[2] Aleš Bokal. "Spletna platforma za analizo hierarhičnih modelov pri odločitvenih problemih", Univerza v Ljubljani Fakulteta za računalništvo in informatiko, 2017.

[3] C. A. Bana e Costa, J. De Corte, J. Vansnick P. "On the Mathematical Foundations of Macbeth", 2004.

[4] C. A. Bana e Costa, J. De Corte, J. Vansnick P. "MACBETH working paper", 2003 London School of Economics and Political Science.

[5] C. A. Bana e Costa, J. Vansnick "The Macbeth Approach: Basic Ideas, Software, And an Application", str. 6–12, 1999.

[6] Catalyze Ltd. HIVIEW3 Starter Guide. Catalyze Ltd, London, 2003.

[7] John S. Hammond, Ralph L. Keeney, Howard Raiffa "The Hidden Traps in Decision Making", Harvard Business School, 1998.

[8] Ralph L. Keeney "Value-Focused Thinking, A Path to Creative Decisionmaking", Harvard University Press, 1992.

[9] Value Tree Analysis, Heisinki University of Technology, Systems Analysis Laboratory [Online]. Dosegljivo: `http://mcda.aalto.fi/value_tree/theory/theory.pdf` [Dostopano 20. 11. 2016].

[10] Valeria Belton, Theodor J. Stewart "Multiple Criteria Decision Analysis An Integrated Approach", str. 52-64, Springer Science+Business Media Dordrecht, 2002.

[11] Javascript in varnost. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/JavaScript#Security. [Dostopano 20. 11. 2016].

[12] Program za MCDA [Online]. Dosegljivo: http://www.catalyzeconsulting.com/software/hiview3/. [Dostopano 20. 11. 2016].

[13] Programski jezik Javascript. [Online]. Dosegljivo: https://www.javascript.com/. [Dostopano 20. 11. 2016].

[14] Oblikovalni jezik. [Online]. Dosegljivo: https://www.w3.org/Style/CSS/. [Dostopano 20. 11. 2016].

[15] Označevalni jezik HTML. [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/HTML. [Dostopano 20. 11. 2016].

[16] Implementacija Blob vmesnika, ki podpira večino brskalnikov. [Online]. Dosegljivo: https://eligrey.com/blog/saving-generated-files-on-the-client-side/. [Dostopano 20. 11. 2016].

[17] Implementacija vmesnika W3C FileSaver, ki podpira večino brskalnikov. [Online]. Dosegljivo: https://github.com/eligrey/FileSaver.js/. [Dostopano 20. 11. 2016].

[18] Implementacija vmesnika W3C Blob, ki podpira večino brskalnikov. [Online]. Dosegljivo: https://github.com/eligrey/Blob.js. [Dostopano 20. 11. 2016].

[19] Knjižnica grafičnih gradnikov jQWidgets. [Online]. Dosegljivo: `https://www.jqwidgets.com/`. [Dostopano 20. 11. 2016].

[20] Interaktivni grafi Highcharts. [Online]. Dosegljivo: `http://www.highcharts.com/`. [Dostopano 20. 11. 2016].

[21] Knjižnica D3.js. [Online]. Dosegljivo: `https://d3js.org/`. [Dostopano 20. 11. 2016].

[22] Orodje SheetJS. [Online]. Dosegljivo: `https://github.com/SheetJS`. [Dostopano 20. 11. 2016].

[23] Knjižnica jsLPSolver. [Online]. Dosegljivo: `https://github.com/JWally/jsLPSolver`. [Dostopano 20. 11. 2016].

[24] Knjižnica algebra.js. [Online]. Dosegljivo: `http://algebra.js.org/`. [Dostopano 20. 11. 2016].

[25] Ukazni načrtovalski vzorec. [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Command_pattern`. [Dostopano 20. 11. 2016].

[26] Ukazni načrtovalski vzorec. [Online]. Dosegljivo: `http://www.dofactory.com/net/command-design-pattern`. [Dostopano 20. 11. 2016].