SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 151

Web-temeljeni sustav za provjeru znanja studenata

Marko Jovanović

Zagreb, siječanj 2011.

Uz najiskrenije zahvale mr.sc. Marku Čupiću
i doc.dr.sc. Domagoju Jakobovići

Sadržaj

1 Uvod

Ubrzanim razvojem u proteklih nekoliko desetljeća računala su uspješno pokorila gotovo sva važnija područja ljudskog djelovanja. Moderno društvo je uvelike postalo ovisno o korištenju računala u svakodnevnim aktivnostima zbog povećanja efikasnosti čovjeka i društva u cjelini. Korištenje računala u edukaciji je utjecalo na unapređenje nastavnog procesa prvenstveno zbog lakše distribucije i prezentacije nastavnih materijala te mogućnosti provedbe ispitivanja uporabom računala.

Tijekom prošlog desetljeća ubrzano su se razvijali sustavi za e-učenje te je njihova implementacija u nastavnom procesu postala rasprostranjena zbog jednostavnosti korištenja i smanjenja troškova. Većina razvijenih sustava za e-učenje sadržava i podsustav za e-ispitivanje. Time se sustavi za e-učenje pokušavaju nametnuti kao najbolja alternativa za klasični nastavni proces koji se ne temelji na uporabi računala. Sustavi za e-ispitivanje najčešće se koriste za provedbu kratkih ispita znanja dok se za složene ispite znanja još uvijek koriste metode bez uporabe računala. Metode ispitivanja bez uporabe računala najjednostavnije se mogu podijeliti na rješavanje ispita na papiru te usmeno ispitivanje. U praksi se te metode kombiniraju kako bi se dobila najbolja procjena ispitanikovog znanja uz što manje utrošenog vremena od strane ispitivača. Zbog oprečnosti navedenih zahtjeva prirodno se upitati postoji li mogućnost provedbe složenog ispitivanja uporabom računala te što je sve potrebno kako bi to ispitivanje bilo barem približno uspješno kao klasične metode ispitivanja. Za provedbu ispitivanja uporabom računala koje bi odgovaralo usmenom ispitivanju nužno je osmisliti način opisa znanja kojeg se ispituje i kojeg u klasičnom ispitivanju posjeduje stručnjak za to područje odnosno ispitivač. Algoritmi za ispitivanje koji simuliraju usmeno ispitivanje ubrajaju se u skupinu algoritama koji se nazivaju algoritmima za adaptivni ispitivanje (*engl. computerized adaptive testing, CAT*) [1].

Algoritmi za adaptivno ispitivanje u svakoj iteraciji postavljaju ispitaniku jedno pitanje ili manju grupu pitanja. Na temelju odgovora ispitanika na pitanja iz trenutne iteracije mijenja se procjena ispitanikovog znanja te se na temelju nove

procjene odabire skup pitanja koji će se postaviti ispitaniku u idućoj iteraciji. Taj proces se ponavlja sve dok nije zadovoljen neki kriterij za prekid ispitivanja poput vremenskog ograničenja za ispitivanje ili maksimalnog broja pitanja koji mogu biti postavljeni ispitaniku.

Teorijski razvoj algoritama za adaptivno ispitivanje je započeo sredinom prošlog stoljeća. Praktične implementacije algoritama za adaptivno ispitivanje nikada nisu zaživjele u popularnim sustavima za e-učenje i e-ispitivanje. Mogući razlozi su složenost tih algoritama i velika količina vremena koju ispitivač mora utrošiti na strukturiranje opisa znanja i stvaranje velike baze pitanja čije je postojanje nužno kako bi algoritam mogao biti efikasan.

U nastavku rada opisana je metoda neadaptivnog ispitivanja uporabom računala te su definirani njezini nedostaci koji su potaknuli razvoj metode adaptivnog ispitivanja. Potom je detaljnije objašnjena sama metoda adaptivnog ispitivanja te način strukturiranja nastavnih cjelina koji se koristi u algoritmu. Postupak implementacije opisan je u zadnjem dijelu rada.

2 Neadaptivno ispitivanje uporabom računala

Ispitivanje uporabom računala u većini slučajeva ima oblik kratke provjere znanja koja se provodi korištenjem određenog sustava za e-ispitivanje. Prilikom pokretanja ispita koji koristi algoritam za neadaptivno ispitivanje iz fonda pitanja se bira skup pitanja unaprijed određene veličine. Kriterij odabira pitanja najčešće ne postoji odnosno pitanja iz fonda se biraju u potpunosti nasumično. Izabrani skup pitanja se prikazuje ispitaniku koji potom upisuje svoja rješenja u sustav te odabire odgovarajuću opciju za vrednovanje upisanih rješenja. Sustav za e-ispitivanje može implementirati razne načine rješavanja ispita. Primjerice, ispitaniku se odabrani skup pitanja može prikazati odjednom ili pak pitanje po pitanje. U drugom slučaju moguće je ispitaniku onemogućiti pristup i mijenjanje rješenja na već prethodno odgovorena pitanja.

Na slici 2-1 prikazan je pseudokod algoritma za nasumičan odabir pitanja bez dodatnih kriterija. Prilikom nasumičnog odabira svako pitanje u fondu ima jednaku vjerojatnost da bude odabrano. Potrebno je naglasiti kako se u algoritmima navedenim u ovom poglavlju pretpostavlja da je veličina fonda pitanja *F* veća ili jednaka broju pitanja *N* koji se odabire.

```
0 nasumicnoGeneriranjeIspita (fond pitanja F,
1                             broj pitanja N){
2     skup nasumično odabranih pitanja Z := {0};
3     dok( |Z|  <  N ){
4          nasumično odaberi pitanje P iz fonda F;
5          ako(P nije u skupu Z)
6               stavi P u skup Z;
7     }
8     vrati Z;
9 }
```

Slika 2-1. Algoritam nasumičnog odabira pitanja bez dodatnih kriterija

Određeni sustavi podupiru oblik ispitivanja u kojem je moguće fond pitanja podijeliti na više međusobno isključivih podgrupa. Prilikom definiranja ispita je moguće odrediti minimalan broj pitanja koje je potrebno izabrati iz svake grupe. Na slici 2-2 prikazan je pseudokod algoritama koji koristi nasumični odabir pitanja bez dodatnih kriterija te uzima u obzir najmanji broj pitanja za svaku podgrupu. Elementi polja podgrupa *G* su cjelobrojne vrijednosti koje predstavljaju minimalan broj pitanja M_i za podgrupu G_i gdje indeks elementa i odgovara indeksu podgrupe G_i. Nakon odabira minimalnog broja pitanja iz svake podgrupe, ostatak pitanja se bira neovisno o grupi kojoj pripada (slika 2-2, petlja na liniji 14).

```
0  nasumicnoGeneriranjeIspita (fond pitanja F,
1                              broj pitanja N,
2                              polje podgrupa G){
3       skup nasumično odabranih pitanja Z := {0};
4       broj podgrupa S := |G|;
5       za (i := 1 do S) {
6            minimalan broj pitanja Mi := G[i];
7            dohvati podgrupu Gi iz fonda F;
8            za (j := 1 do Mi){
9                 nasumično odaberi pitanje P iz fonda Gi;
10                ako(P nije u skupu Z)
11                     stavi P u skup Z;
12           }
13      }
14      dok( |Z|  <  N ){
15           nasumično odaberi pitanje P iz fonda F;
16           ako(P nije u skupu Z)
17                stavi P u skup Z;
18      }
19      vrati Z;
20 }
```

Slika 2-2 Algoritam nasumičnog odabira pitanja bez dodatnih kriterija uz raspodjelu po podgrupama

U osnovnom obliku neadaptivnog ispitivanja svako pitanje iz fonda ima jednaku vjerojatnost da bude odabrano. U obzir se ne uzimaju kriteriji poput težine pitanja ili prosječnog vremena potrebnog za odgovaranje na pitanje. Zbog toga postoji mogućnost da generirani ispit ima neprimjerenu težinu ili je pak za njegovo rješavanje potrebno više vremena nego što ga ispitanik ima na raspolaganju. U navedenim slučajevima je velika vjerojatnost da konačna procjena ispitanikovog znanja o zadanom području neće ni približno odgovarati stvarnoj razini znanja.

2.1 Korištenje dodatnih kriterija prilikom nasumičnog odabira pitanja

Težina ispita najčešće direktno utječe na vrijeme potrebno za rješavanje generiranog ispita. Svako ispitivanje ograničeno je maksimalnim trajanjem. Stvarno vrijeme potrebno za rješavanje ispita može prelaziti granicu maksimalnog trajanja ispita u slučaju kada odabrana pitanja tvore iznadprosječno težak ispit. U tom slučaju ispitanik neće imati dovoljno vremena na raspolaganju što će utjecati na njegov konačan rezultat.

Fond pitanja može sadržavati pitanja niske težine koja zahtijevaju veću količinu vremena za rješavanje. Primjerice, to mogu biti pitanja u kojima se provode jednostavne matematičke operacije, ali zahtijevaju puno međurezultata i računanja. Metoda nasumičnog odabira pitanja u kojoj se kao kriterij ne koristi prosječno vrijeme potrebno za odgovaranje na pitanje ima određene nedostatke. Sustav koji ne uzima u obzir kriterij prosječnog vremena može odabrati skup pitanja koji u cjelini nije težak za rješavanje, ali čije stvarno vrijeme rješavanja prelazi zadano maksimalno trajanje ispitivanja.

Algoritam nasumičnog odabira pitanja može se unaprijediti uvođenjem kriterija koji će utjecati na vjerojatnost odabira pitanja. Primjerice, kriteriji mogu biti težina pitanja i prosječno vrijeme potrebno za odgovaranje na pitanje.

2.1.1 Kriterij težine pitanja

Težinu pitanja T_i određuje autor pitanja prilikom unosa u sustav za ispitivanje. Najčešće je to prirodan broj kojeg autor proizvoljno određuje uzimajući u obzir usporedbu novih pitanja s već postojećima u sustavu. Kriterij težine se može iskoristiti za smanjivanje vjerojatnosti generiranja ispita koji su prejednostavni ili pak pretjerano teški. Definira se mjera srednje težine S_t kao aritmetička sredina svih težina pitanja iz zadanog fonda. Za svako pitanje izračuna se M_i kao modul razlike srednje težine S_t i težine T_i :

$$M_i = | S_t - T_i |$$

Pitanje *i* ima veću vjerojatnost da bude odabrano od pitanja *j* ako:

$$M_i < M_j .$$

U algoritmu nasumičnog odabira pitanja na temelju kriterija težine (slika 2-3) koristi se funkcija *nasumicniOdabirPitanjaPoTezini*. Funkcija za svako pitanje izračuna vrijednost M_i te u ovisnosti o toj vrijednosti dodjeljuje pitanju određenu vjerojatnost. Nakon toga provodi se nasumičan odabir jednog pitanja uzimajući u obzir izračunate vjerojatnosti. Funkciji se kao argument predaje srednja težina S_t.

```
0   nasumicnoGeneriranjeIspita(fond pitanja F,
1                              broj pitanja N){
2       skup nasumično odabranih pitanja Z = {0};
3       izračunati srednju težinu St;
4       dok( |Z|  <  N ){
5           pitanje P = nasumicniOdabirPitanjaPoTezini(St);
6           ako(P nije u skupu Z)
7           stavi P u skup Z;
8       }
9       vrati Z;
10  }
```

Slika 2-3 Algoritam nasumičnog odabira pitanja na temelju kriterija težine pitanja

Određivanje težine T_i prilikom stvaranja novog pitanja može predstavljati problem jer težina nema unaprijed određenu mjeru. Svaki autor pitanja ima različit kriterij vrednovanja težine T_i. Autor pitanja bi trebao usporediti novo pitanja s velikim brojem postojećih pitanja iz sustava kako bi u ovisnosti o postojećim težinama dobro odredio težinu novog pitanja. Taj proces bi zahtijevao veliku količinu vremena što ima za posljedicu da se navedeni način ne može primijeniti u praksi. U slučaju da težine pitanja nisu dobro određene, smanjila bi se efikasnost sustava. Prvi korak ka savladavanju problema je ograničavanje vrijednosti težine T_i na manji interval prirodnih brojeva, primjerice $1 \leq Ti \leq 10$. Utjecaj subjektivnosti autora na određivanje težine mogao bi se smanjiti prethodnim dogovorom svih autora o kriterijima težinskog vrednovanja pitanja.

2.1.2 Kriterij prosječnog vremena za odgovaranje na pitanje

Prosječno vrijeme V_i potrebno da ispitanik odgovori na pitanje *i* također može biti kriterij pri odabiru pitanja. Autor određuje inicijalno vrijeme V_i prilikom unosa pitanja u sustav za ispitivanje. Iznos V_i mogao bi se mijenja nakon svakog ispitivanja ispravljajući prethodnu vrijednost u ovisnosti o vremenu koje je potrošeno na pitanje *i* tijekom zadnjeg ispitivanja. Navedeni postupak omogućava da vrijednost V_i poprimi trajanje koje će odgovarati stvarnom prosječnom vremenu potrebnom za odgovaranje na pitanje. U slučaju da taj postupak nije implementiran, može se dogoditi da autor pitanja postavi prekratko ili predugo prosječno vrijeme u usporedbi sa stvarnim prosječnim vremenom. Prosječno vrijeme V_i za razliku od težine T_i ima definiranu mjeru odnosno ima definirano trajanje u sekundama. Zbog toga je kriterij prosječnog vremena V_i lakši za implementaciju od kriterija težine pitanja. Prosječno vrijeme V_i se može korigirati prema formuli:

$$V_i = \frac{V_{iP} \times n + \sum_{k=1}^{n} V_{ik}}{2 \times n}$$

Parametar V_{iP} je vrijednost prosječnog vremena odgovaranja na pitanje *i* prije početka ispitivanja. Parametri V_{ik} su vremena koja su studenti potrošili za odgovaranje na pitanje *i*. U obzir se uzimaju svi točni odgovori i samo netočni odgovori za koje vrijedi da V_{ik} nije manje od V_{iP}. Broj takvih odgovora je pohranjen u parametru *n.* Navedena korekcija se obavlja nakon što je završeno ispitivanje odnosno kada su svi ispitanici odgovorili na pitanja. Množenjem početnog prosječnog vremena V_{iP} i broja odgovora *n* sprječava se veći utjecaj samo jednog ispitivanja na vrijednost V_i.

Definira se mjera srednjeg prosječnog vremena S_v kao aritmetička sredina svih prosječnih vremena V_i pitanja iz zadanog fonda pitanja *F*. Za svako pitanje se izračuna C_i kao modul razlike srednjeg prosječnog vremena S_v i prosječnog vremena V_i:

$$C_i = | S_v - V_i |$$

Pitanje *i* ima veću vjerojatnost da bude odabrano od pitanja *j* ako:

$$C_i < C_j$$

.

```
0   nasumicnoGeneriranjeIspita(fond pitanja F,
1                              broj pitanja N){
2       skup nasumično odabranih pitanja Z = {0};
3       izračunati srednje prosječno vrijeme S_v;
4       dok( |Z|  <  N ){
5           pitanje P =
6               nasumicniOdabirPitanjaPoVremenu(S_v);
7           ako(P nije u skupu Z)
8               stavi P u skup Z;
9       }
10      vrati Z;
11  }
```

Slika 2-4 Algoritam nasumičnog odabira pitanja na temelju kriterija prosječnog vremena potrebnog za odgovaranje na pitanje

U algoritmu nasumičnog odabira pitanja na temelju kriterija prosječnog vremena potrebnog za odgovaranje na pitanje (slika 2-4) koristi se funkcija `nasumicniOdabirPitanjaPoVremenu`. Funkcija za svako pitanje izračuna vrijednost C_i te u ovisnosti o toj vrijednosti pitanju pridijeli određenu vjerojatnost. Nakon toga provodi se nasumičan odabir jednog pitanja uzimajući u obzir izračunate vjerojatnosti. Funkciji se kao argument predaje srednja težina S_v.

2.1.3 Kriterij težine pitanja i prosječnog vremena za odgovaranje na pitanje

Algoritam nasumičnog odabira se može unaprijediti istovremenim korištenjem oba prethodno opisana kriterija, prosječnog vremena za odgovaranje na pitanje V_i i težine pitanja T_i kao kriterija za odabir pitanja. Za svako pitanje iz

fonda izračunava se vjerojatnost u ovisnosti o V_i i T_i te izračunatim vrijednostima S_v i S_t prethodno opisanim u poglavlju.

Funkcija `nasumicniOdabirPitanjaPoVremenuITezini` nasumično odabire pitanje uzimajući u obzir izračunatu vjerojatnosti svakog pitanja (slika 2-5).

```
0   nasumicnoGeneriranjeIspita(fond pitanja F,
1                              broj pitanja N){
2       skup nasumično odabranih pitanja Z = {0};
3       izračunati srednje prosječno vrijeme Sv;
4       izračunati srednju težinu St;
5       dok( |Z|  <  N ){
6          pitanje P =
7            nasumicniOdabirPitanjaPoVremenuITezini(Sv, St);
8          ako(P nije u skupu Z)
9            stavi P u skup Z;
10      }
11      vrati Z;
12  }
```

Slika 2-5 Algoritam nasumičnog odabira pitanja na temelju kriterija prosječnog vremena za odgovaranje na pitanje i težine pitanja

3 Adaptivno ispitivanje uporabom računala

Algoritmi za adaptivno ispitivanje razvijaju se s ciljem dobivanja preciznije procjene ispitanikovog znanja od procjene koja se dobiva korištenjem algoritama za neadaptivnog ispitivanje. To se pokušava postići modeliranjem iterativnog algoritma koji u svakoj iteraciji određuje koje je optimalno pitanje ili manji skup pitanja za iduću iteraciju na temelju odgovora na sva prethodno postavljena pitanja. Pod pojmom optimalno pitanje podrazumijeva se pitanje čijim bi se odgovorom mogla najbolje modificirati trenutna procjena znanja kako bi bila što točnija u usporedbi sa stvarnom razinom znanja studenta. Navedene metode ispitivanja se nazivaju adaptivnima zbog postupka prilagođavanja ispitivanja u ovisnosti o znanju kojeg ispitanik prezentira tijekom odgovaranja na odabrani slijed pitanja.

Iz prethodnog objašnjenja lako je uočiti glavne korake adaptivnog ispitivanja:

1. odabir optimalnog pitanja pomoću određene metode na temelju trenutne procjene ispitanikovog znanja,
2. prezentiranje odabranog pitanja ili manjeg skupa pitanja ispitaniku te prihvat ispitanikovog odgovora,
3. promjena procjene znanja studenta na temelju prethodno odgovorenih pitanja,
4. ponavljanje koraka 1-3 dok nije zadovoljen kriterij zaustavljanja.

Za odabir optimalnog pitanja ili manjeg skupa pitanja u svakom koraku se koriste razne metode od kojih je najpoznatija *teorija odgovora na čestice* (engl. *item response theory, IRT*) [1]. Navedena teorija je paradigma za osmišljavanje, provođenje i analizu raznih oblika ispitivanja. Temelji se na pretpostavci da se vjerojatnost točnog odgovora na pitanje može modelirati funkcijom vjerojatnosti koja ovisi o ispitanikovim mogućnostima i svojstvima dotičnog pitanja. Određivanje te funkcije zahtjeva kalibriranje fonda pitanja pri svakoj izmjeni tog fonda odnosno kod dodavanja novih pitanja u fond. Time se utvrđuju parametri za cjelokupni skup pitanja što se koristi prilikom ispitivanja za modifikaciju procjene znanja ispitanika. Jedan od parametara je i vrijednost koja utječe na vjerojatnost točnog odgovora na

neko pitanje s ciljem prepoznavanja točnih odgovora kojeg je ispitanik nasumično odabrao.

Proces kalibriranja se temelji na provođenju velikog broja probnih ispitivanja koja sadržavaju nova pitanja što je jedna od glavnih mana adaptivnog ispitivanja temeljenog na *IRT*-u. Kalibriranje je moguće provesti i na način da se određeni broj novih pitanja postavlja tijekom redovnog ispitivanja te se na temelju dobivenih odgovora na ta pitanja pokušava optimizirati funkcija vjerojatnosti. Ta metoda također ima nedostatke jer se na taj način određivanje funkcije vjerojatnosti usporava u odnosu na metodu koja za kalibraciju koristi probna ispitivanja. Također, postoji i mogućnost da će korištenje slabo kalibriranog fonda pitanja koji sadrži nova pitanja utjecati na valjanost procjene ispitanikovog znanja. Iz tog razloga se odgovori na nova pitanja ne koriste za promjenu procjene znanja već samo za kalibriranje funkcije vjerojatnosti. Ignoriranje novih pitanja u procjeni znanja smanjuje broj prethodno kalibriranih pitanja koja mogu biti postavljena tijekom ispitivanja i koja se koriste za promjenu procjene znanja studenta. To utječe na točnost te procjene jer je se smatra da točnost raste s brojem postavljenih pitanja. Kalibriranje fonda pitanja predstavlja glavni nedostatak tog tipa adaptivnog ispitivanja i izravno je povezano s područjima primjene metode jer je potrebno uložiti puno vremena za stvaranje velike baze pitanja i za kalibriranje tih zadataka. Zbog toga sustavi koji implementiraju tu metodu se u većini slučajeva razvijaju i koriste za ispitivanje velikog broja ispitanika te se svode na velike sustave poput sustava za provedbu prijemnih ispita na sveučilišta u Sjedinjenim Američkim Državama [2].

Sustavi za adaptivno ispitivanje se često koriste za klasifikaciju ispitanika u određene grupe na temelju njihovog znanja. Taj način ispitivanja se naziva i *klasifikacijsko ispitivanje uporabom računala* (engl. *computerized classification test*) [3]. Najjednostavniji primjer klasifikacije je klasifikacija u dvije grupe od kojih jedna ubraja ispitanike koji ne posjeduju dovoljno znanja o nekom području dok se u drugoj skupini nalaze svi studenti koji su tijekom ispitivanja pokazali da se njihovo znanje nalazi na zadovoljavajućoj razini. Druga skupa se može podijeliti na nekoliko manjih skupina. Primjerice, moguće je razlikovati ispitanike koji posjeduju samo osnovno znanje od onih ispitanika koji su pokazali visoku razinu znanja iz određenog područja. Ispitivanje će duže trajati ako se ispitanik nalazi na granici

između dva razreda jer će biti potrebno više pitanja kako bi se točno odredilo kojem razredu pripada. U suprotnom, ispitivanje je kraće kada se ispitanik ne nalazi blizu granice između dvije klase jer za rezultat ispitivanja nije potrebno odrediti točnu procjenu znanja već samo klasu kojoj taj ispitanik pripada.

Način odabira optimalnog pitanja može predstavljati slabu točku u provedbi adaptivnog ispitivanja u slučaju kada se određena pitanja biraju jako često za veći broj studenata sa približno jednakom razinom znanja. U tom slučaju je moguće da ispitanici unaprijed doznaju koja su česta pitanja od ispitanika koji su već riješili taj ispit. Taj problem moguće je prevladati dodavanjem sigurnosnog mehanizma kojim bi se pratio broj prikazivanja svakog zadatka u svim ispitivanjima te koji bi zabranio korištenje određenog zadatka u slučaju da je on već bio iskorišten određeni broj puta. Umjesto odabira pitanja koji sadrži najveću informaciju u svakoj iteraciji, moguće je koristiti razne metode za odabir idućeg pitanja na temelju vjerojatnosti koja je proporcionalna informaciji svakog pitanja. Time bi se reducirao odabir pitanja koja su često optimalna jer bi postojala mogućnost odabira zadataka koji su blizu optimalnom pitanju i koji bi bili rjeđe prikazivani ispitaniku u slučaju da se preferira samo optimalno pitanje.

Kriteriji zaustavljanja ispitivanja se razlikuju u ovisnosti o algoritmu koji se koristi za provedbu adaptivnog ispitivanja. Najčešće je nužno definirati samo maksimalno trajanje ispita, ali neki sustavi za e-ispitivanje zahtijevaju i podatke poput minimalnog trajanja te minimalnog i maksimalnog broja pitanja koji može biti prikazan ispitaniku.

Pregled i izmjena prethodnih odgovora je onemogućena jer bi se u suprotnom narušila konzistentnost ostvarene procjene ispitanikovog znanja. To je direktna posljedica načina na koji se provodi adaptivno ispitivanje jer je trenutno stanje procjene znanja rezultat odgovora na prethodno postavljeni slijed pitanja. Izmjena odgovora na bilo koje pitanja iz toga slijeda bi gotovo uvijek povlačila potrebu za postavljanjem novog slijeda pitanja poslije izmijenjenog pitanja jer bi se procjena znanja ispitanika promijenila na temelju te izmjene. Slijed pitanja poslije izmijenjenog odgovora ne bi odgovarao slijedu pitanja koji bi bio postavljen da je to pitanje imalo taj isti odgovor prilikom prvog odgovaranja. Određene metode za adaptivno ispitivanje poput *teorije odgovora na čestice* za sljedeće pitanje nakon

pogrešnog odgovora na već postavljeno pitanje dodjeljuju lakše pitanje od prethodnoga postavljenog pitanja. Ispitanik će na rješavanje dobiti jako lagan ispit ako na niz pitanja odgovori netočno. Ta činjenica bi se mogla zloupotrijebiti od strane ispitanika u slučaju da on ima mogućnost izmjene prethodno odgovorenih pitanja. Primjerice, ispitanik bi mogao zaključiti da je na prethodno odgovorene probleme dao netočan odgovor ako uoči da se težina novih zadataka smanjuje. Također, ispitanik može namjerno odgovoriti krivo na uvodni slijed pitanja čime bi u kasnijoj fazi ispitivanja dobio jednostavnija pitanja, a na kraju bi se mogao vratiti na prethodna pitanja i korigirati krive odgovore.

Implementacija adaptivnog ispitivanja na temelju *teorije odgovora na čestice se* ne isplati za manje sustave zbog prethodno opisanog postupka kalibracije koji je nužan za uspješno provođenje ispitivanja. Iz toga razloga je potrebno osmisliti alternativne metode na temelju kojih bi se biralo optimalno pitanje u svakoj iteraciji. Algoritam za adaptivno ispitivanje implementiran i objašnjen u ovom radu temelji se na iskorištavanju semantičkih veza definiranih u strukturiranom hijerarhijskom prikazu nastavnih cjelina koji će biti objašnjen u idućem poglavlju. Na taj način pokušava se izbjeći potreba za kalibracijom fonda pitanja i time omogućiti korištenje adaptivnog ispitivanja na kolegijima s manjim brojem studenata na kojima je potrebno izvođenje kraćih provjera znanja. Jednom stvoreni fond pitanja bi se mogao koristiti godinama uz moguće manje promjene fonda koje ne bi zahtijevale veliki utrošak vremena.

U sljedećem poglavlju je objašnjen način strukturiranja nastavnih cjelina koji se koristi u kasnije opisanom algoritmu za adaptivno ispitivanje te je opisana veza između zadatak i nastavnih cjelina koje ispituju pojedini zadaci.

4 Strukturiranje nastavnih cjelina

Jedan od nužnih uvjeta za efikasnu provedbu ovog tipa adaptivnog ispitivanja je strukturirani opis nastavnih cjelina na temelju kojeg će algoritam za adaptivno ispitivanje donositi odluke o odabiru pitanja u svakoj iteraciji algoritma. Elementi takve strukture moraju sadržavati semantičke veze između nastavnih cjelina odnosno potrebno je stvoriti semantičku mrežu [4] koja će definirati odnose između nastavnih elemenata.

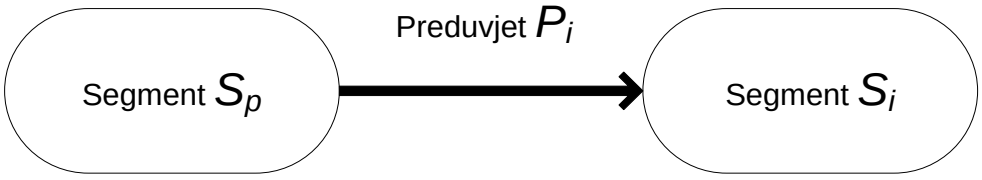
Sustavi za e-ispitivanje su najčešće podsustavi sustava za e-učenje pa se kao početna točka analize postojećih rješenja nametnulo proučavanje normi i standarda za opisivanja objekata za učenje u sustavima za e-učenje. U sklopu ovog diplomskog rada proučeni su norme LOM [5] i SCORM [6]. Navedene norme su definirane s ciljem da standardiziraju pohranjivanje nastavnih sadržaja na jednostavan i svima razumljiv način. Dodatan kriterij je pružanje pohrane širokog spektra podataka koje bi sustavi za e-učenje mogli zatrebati. Nije teško učiti kako su navedena dva zahtjeva oprečna. Primjerice, norma LOM verzije 1.0 definirana u IEEE 1484.12.1 pruža mogućnost opisa nastavnog elementa koristeći oko 40 atributa grupiranih u 9 kategorija. Vidljivo je kako bi tako opširno opisivanje svakog nastavnog elementa na nekom predmetu zahtijevalo izrazito veliku količinu vremena. Nužno je napomenuti kako u LOM normi pojedine atribute nije potrebno unositi no zaključeno je kako bi i u tom slučaju količina utrošenog vremena bila prevelika za praktičnu upotrebu. Potrebno je uočiti i činjenicu da su prethodno navedene norme definirane na području strukturiranja nastavnih cjelina za učenje te ne sadržavaju elemente za definiranje podataka koji su nužni u algoritmima za adaptivno ispitivanje.

Prethodno navedene norme LOM i SCORM uvelike su utjecale na model podataka koji će biti definiran dalje u tekstu. U nastavku poglavlja je opisan način strukturiranja nastavnih cjelina koji se koristi u kasnije objašnjenom algoritmu za adaptivno ispitivanje.

4.1 Predloženi način strukturiranja nastavnih cjelina

Sva područja u obrazovanju sastavljena su od nastavnih cjelina. Jedna nastavna cjelina se može podijeliti na manje cjeline odnosno teme. Teme se potom mogu podijeliti na manje podteme i taj proces je moguće ponavljati sve dok se ne rafiniraju elementi koji nisu sastavljeni od manjih cjelina. Takvi elementi predstavljaju atomarnu jedinicu znanja odnosno osnovne koncepte iz nastavnog područja. Iz prethodnog objašnjenja je moguće uočiti kako se za strukturirani opis nastavnih cjelina može koristiti stablo s dvije vrste čvorova, temama i konceptima. Teme su unutrašnji čvorovi stabla koji mogu sadržavati druge čvorove, neovisno o vrsti tih čvorova. Koncepti su nedjeljivi elementi znanja koji su grupirani u teme te mogu biti samo listovi stabla. Prethodno opisani tipovi čvorova zajednički se nazivaju segmentima znanja.

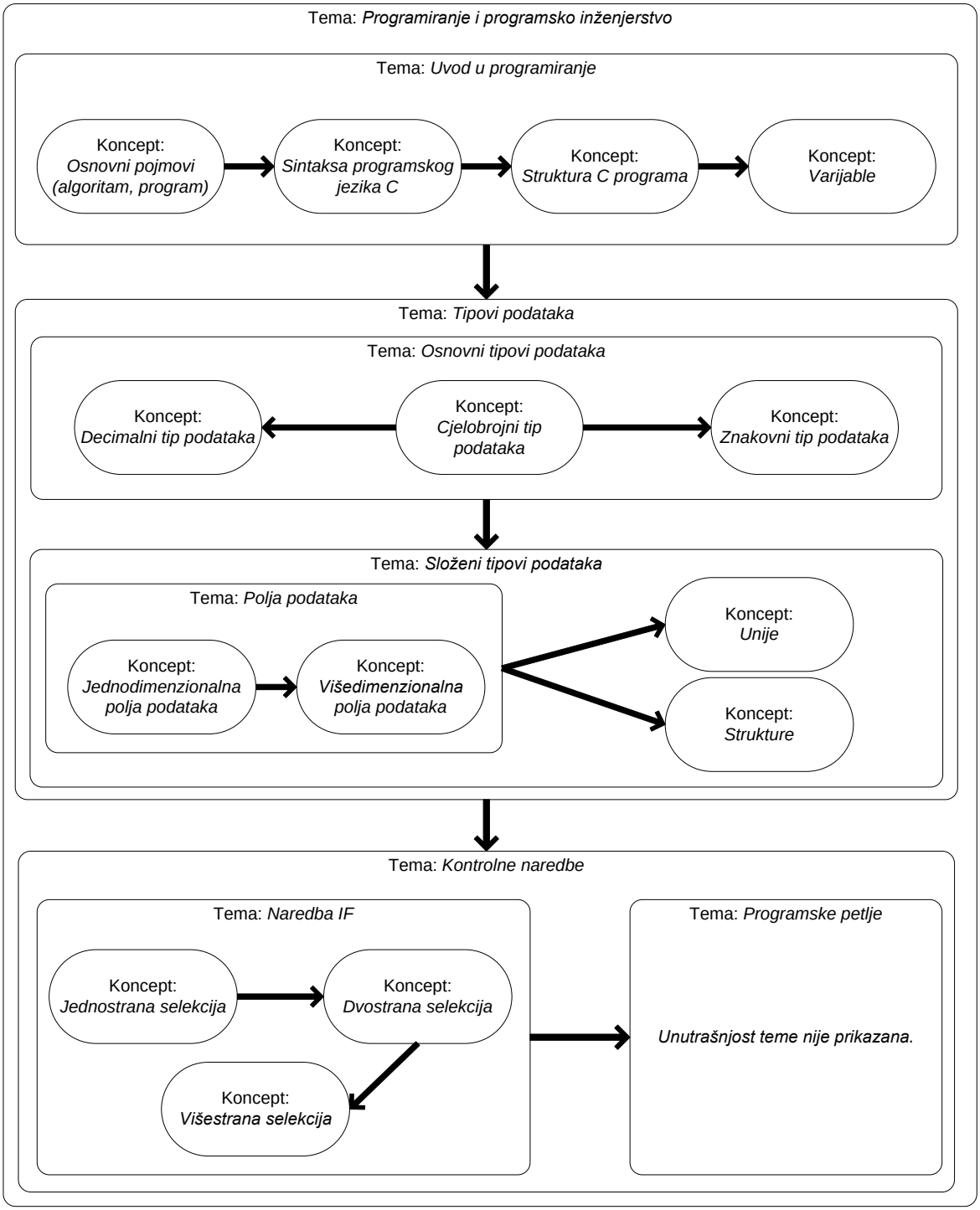
Svaki segment znanja se nalazi u složenim međuovisnostima o drugim segmentima u istom stablu. Prva vrsta međuovisnosti je prethodno opisana raspodjela segmenata u teme. Drugu vrstu međuovisnosti je moguće opisati kao količinu znanja o ostalim segmentima koju student mora imati kako bi uspješno naučio neki drugi segment znanja. Ta vrsta međuovisnosti nazvana je preduvjet segmenta. Svaki preduvjet nekog segmenta S_i je predstavljen kao postotak znanja o nekom segmentu S_p koji student mora posjedovati kako bi mogao svladati segment S_i. Preduvjet P_i moguće je grafički prikazati kao brid u grafu usmjeren od segmenta koji je preduvjet S_p prema segmentu S_i (slika 4-1).



Slika 4-1. Grafički prikaz preduvjeta

Navedeni pojmovi biti će dodatno pojašnjeni na primjeru predmeta *Programiranje i programsko inženjerstvo* na slici 4-2. Na slici je prikazano samo prvih nekoliko poglavlja koji se obrađuju na predmetu. Za svaki segment u prikazu naveden je tip te naziv segmenta. Segmenti su grupirani u teme pa su primjerice

teme *Uvod u programiranje*, *Tipovi podataka* i *Kontrolne naredbe* nalaze unutar teme *Programiranje i programsko inženjerstvo* koja ujedno predstavlja i sam kolegij.



Slika 4-2 Primjer dekompozicije predmeta Programiranje i programsko inženjerstvo

U grafu segmenata i preduvjeta moguće je koristiti četiri vrste preduvjeta koje su definirane u ovisnosti o tipovima segmenata koje taj preduvjet povezuje:

- preduvjet *koncept → koncept*: najosnovniji tip preduvjeta kojim je izravno definirana ovisnost jednog koncepta o drugome konceptu; primjer ovog tipa preduvjeta je preduvjet između koncepata *Jednodimenzionalna polja podataka* i *Višedimenzionalna polja podataka* koji predstavlja nužnost poznavanja prvog koncepta kako bi postojala mogućnost uspješno savladavanja drugog koncepta (slika 4-2);

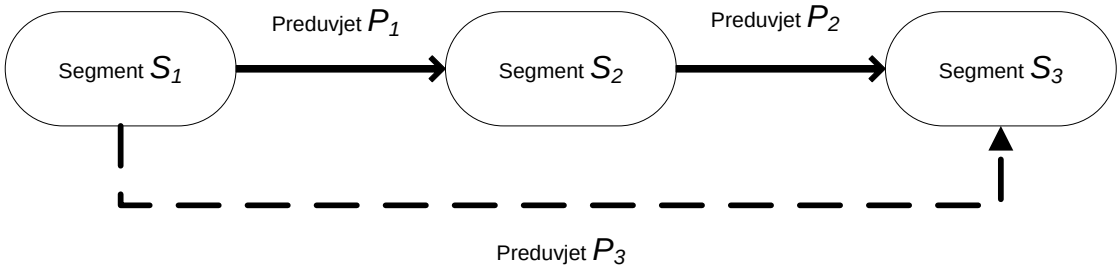
- preduvjet *koncept → tema*: ovaj tip preduvjeta definira potrebu za posjedovanjem određenom razinom znanja o *konceptu* kako bi postojala mogućnost uspješnog učenja bilo kojeg segmenta koji se nalazi u *temi*;

- preduvjet *tema → koncept*: ovaj tip preduvjeta definira potrebu za posjedovanjem određene razine znanja o svim segmentima u *temi* kako bi postojala mogućnost uspješnog učenja *koncepta*; primjer ovog tipa preduvjeta je preduvjet između teme *Polja podataka* i koncepta *Strukture* koji predstavlja nužnost poznavanja oba koncepta iz teme *Polja podataka* jer u protivnome nije moguće kvalitetno naučiti koncept *Strukture* (slika 4-2);

- preduvjet *tema → tema*: ovaj tip preduvjeta predstavlja najkompliciraniju vrstu preduvjeta jer definira potrebu za posjedovanjem određene razine znanja o svim segmentima u *prvoj temi* kako bi postojala mogućnost uspješnog učenja bilo kojeg segmenata iz *druge teme*; primjer ovog tipa preduvjeta je preduvjet između teme *Osnovni tipovi podataka* i teme *Složeni tipovi podataka* koji predstavlja nužnost poznavanja koncepata iz prve teme tj. razumijevanje osnovnih tipova podataka kako bi bilo moguće razumijeti koncepte iz druge teme koji predstavljaju znanje o složenim tipovima podataka (slika 4-2).

Moguće je uočiti kako preduvjeti i segmenti formiraju usmjereni aciklički graf što će biti iskorišteno u kasnije definiranom algoritmu za adaptivno ispitivanje. Nepostojanje ciklusa se može objasniti time što nije moguće da neki skup segmenata ciklički ovise jedan o drugome zbog procesa učenja pomoću kojeg čovjek savladava nova znanja. Taj proces se temelji na iskorištavanju postojećeg znanja odnosno prethodno naučenih segmenata znanja kako bi se naučili novi segmenti znanja te spada u osnovne postulate edukacije.

Potrebno je uočiti kako relacija preduvjeta ima osobinu tranzitivnosti. Definiraju se preduvjet P_1 koji povezuje segmente S_1 i S_2 na način da je segment S_1 preduvjet segmenta S_2 te preduvjet P_2 koji povezuje segmente S_2 i S_3 na način da je segment S_2 preduvjet segmenta S_3. Tvrdnja je da se tada može definirati novi preduvjet P_3 koji povezuje segmente S_1 i S_3 na način da je segment S_1 preduvjet segmentu S_3 (slika 4-3).



Slika 4-3. Tranzitivnost relacije preduvjeta

Dokaz prethodno navedene tvrdnje može se pronaći u definiciji preduvjeta P_2 koja kaže da je segment S_2 preduvjet nekog drugog segmenta S_3 što znači da je potrebno određeno poznavanje segmenta S_2 za uspješno savladavanje segmenta S_3. Ako je definiran i preduvjet P_1 vidljivo je da je za savladavanje segmenta S_2 nužno znanje o S_1. Kako bi se naučio S_3 prvo se mora naučiti S_2 koji dalje ovisi o poznavanju segmenta S_1 stoga je očito da segment S_3 indirektno ovisi o segmentu S_1 te je moguće definirati novi preduvjet koji predstavlja implicitni odnos između S_1 i S_3. Tu činjenicu je moguće iskoristiti za pojednostavljivanje grafa preduvjeta jer nije potrebno definirati sve moguće preduvjete za zadani skup

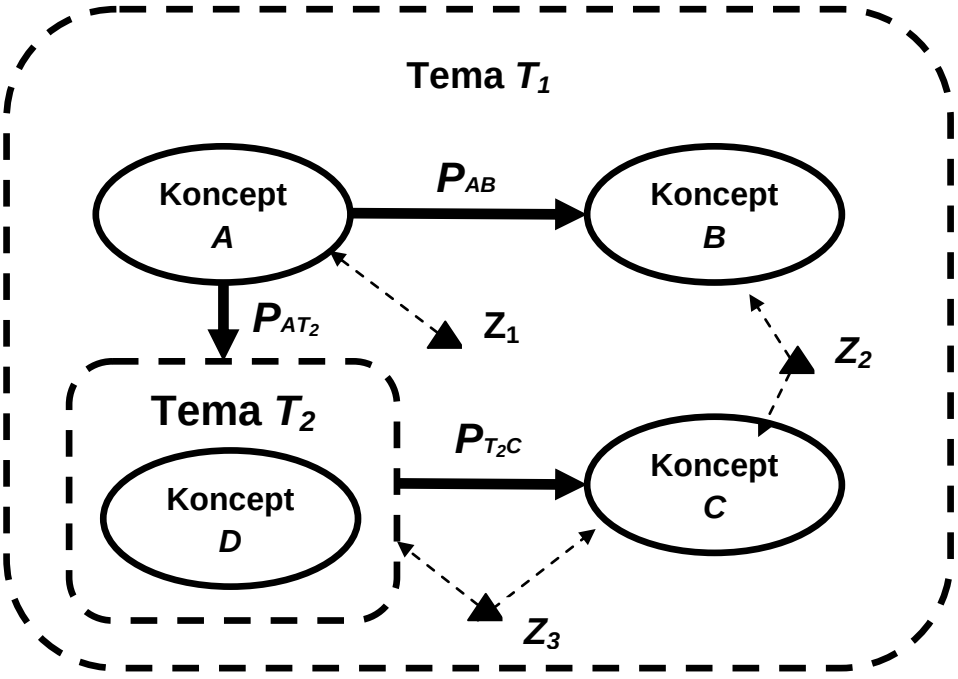
segmenata već se neki preduvjeti mogu indirektno uočiti na temelju tranzitivnosti relacije preduvjeta.

Kao jednostavan primjer pojednostavljivanja može se iskoristiti prikaz dekompozicije predmeta *Programiranje i programsko inženjerstvo* sa slike 4-2. U navedenom prikazu nije definiran preduvjet kojim bi se jasno naglasila nužnost poznavanja teme *Uvod u programiranje* za uspješno savladavanje teme *Kontrolne naredbe* iako je poznavanje pojedinih koncepata iz prve teme, poput koncepta *Varijable*, nužno za shvaćanje složenijih koncepata u *Kontrolnim naredbama*. Taj odnos indirektno je definiran pomoću dva preduvjeta; tema *Uvod u programiranje* je preduvjet temi *Tipovi podataka* koja je dalje preduvjet temi *Kontrolne naredbe*.

4.2 Veza između nastavnih cjelina i ispitnih pitanja

U hijerarhiji nastavnih cjelina definiran je i element koji se predstavlja ispitna pitanja. Svako pitanje sadrži popis koncepata i tema koje to pitanje pokriva. Nije definirana mjera kojom se može odrediti pojedina važnost koncepta ili teme u pitanju jer bi određivanje tih podataka dodatno usporilo izradu baze pitanja. Iz navedenog razloga svaki koncept u pitanju ima jednaku važnost. U slučaju kada je pitanje vezano uz temu svi koncepti iz te teme se tretiraju kao da su povezani s pitanjem.

Na slici 4-4 je prikazan jednostavan primjer hijerarhije nastavnih cjelina i ispitnih pitanja. Skup ispitnih pitanja odnosno zadataka {Z_1, Z_2, Z_3} povezan je isprekidanim strelicama s temama i konceptima na koje se svaki zadatak odnosi. Kako bi se uspješno savladao koncept *B* potrebno je uspješno naučiti koncept *A*. Taj odnos je definiran preduvjetom prikazanim kao puna strelica od koncepta A do koncepta *B* sa zadanom težinom preduvjeta P_{AB}. Jednako vrijedi i za odnos koncepta *A* i teme T_2 kao i za temu T_2 i koncept *C*. Vidljivo je da zadatak Z_3 ispituje temu T_2 i koncept *C*, zadatak Z_2 ispituje koncepte *B* i *C* dok zadatak Z_1 ispituje samo koncept *A*.



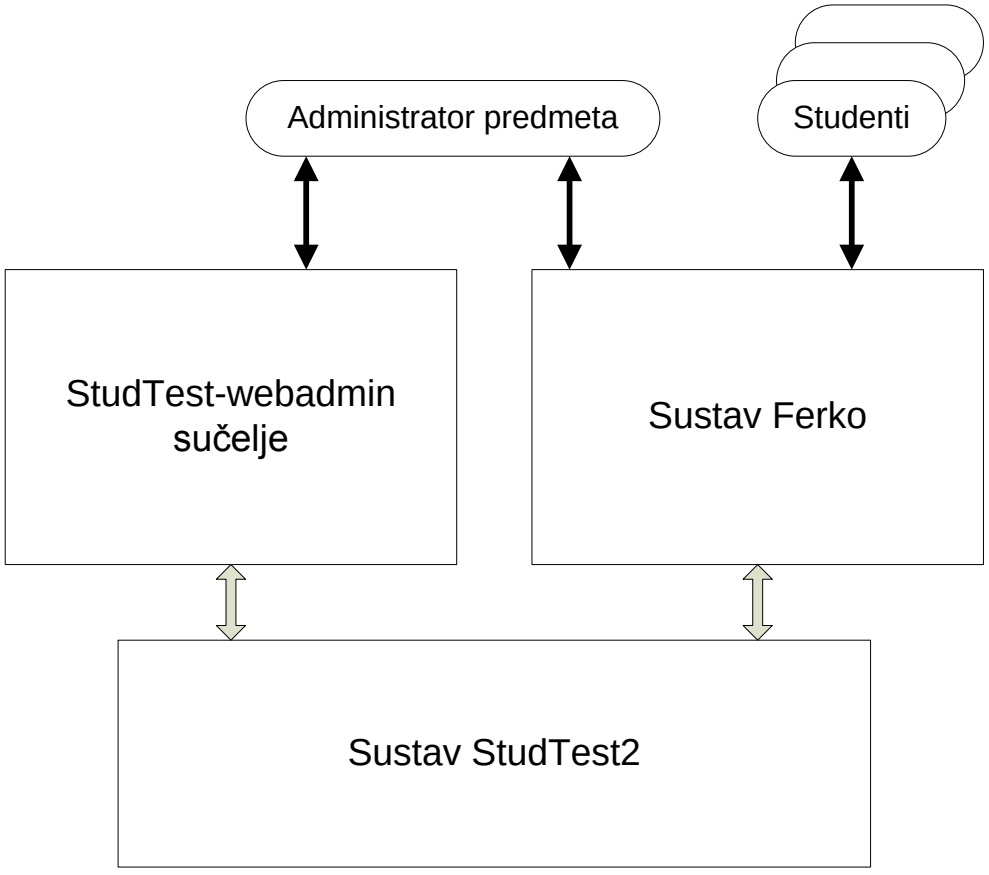
Slika 4-4. Prikaz povezivanja zadatka i segmenata znanja

5 Opis implementacije adaptivnog ispitivanja

U sklopu ovog diplomskog rada u sustave *Studtest2* [7] i *Ferko* [8] su implementirane funkcionalnosti koje podupiru adaptivno ispitivanje temeljeno na strukturiranoj hijerarhiji nastavnih cjelina.

Sustav *Studtest2* je složen sustav za e-ispitivanje i pruža mnoštvo mogućnosti za korištenje. Struktura sustava omogućava jednostavnu nadogradnju zbog razdvojenosti na komponente od kojih je svaka zadužena za mali dio funkcionalnosti.

Sustav *Ferko* je sustav za upravljanje kolegijima koji ujedno služi i kao sučelje za prikaz ispitivanja koje provodi *StudTest2* sustav.



Slika 5-1 Uporaba sustava StudTest2 i Ferko

Na slici 5-1 prikazana je uporaba sustava *StudTest2* i *Ferko*. Administrator predmeta koristi *StudTest-webadmin* sučelje kako bi upravljao podacima u sustavu *StudTest2*. Kroz navedeno sučelje obavljaju se svi poslovi vezani uz

administriranje ispita i zadataka. Administrator kroz sustav *Ferko* upravlja pristupom testovima i sučeljem za stvaranje strukturirane hijerarhije nastavnih cjelina odnosno stabla segmenata. Studenti kroz sustav *Ferko* mogu pristupati rješavanju ispita.

Oba sustava su implementirana u programskom jeziku *Java* te se oslanjaju na svojstva tog objektno orijentiranog programskog jezika kako bi se modelirali koncepti na kojima se temelji administracija kolegija i e-ispitivanje.

Sustavi koriste biblioteku *Hibernate* [9] za savladavanje objektno-relacijskog preslikavanja koji predstavlja glavni problem u istovremenom korištenju objektnog modela podataka i relacijske baze podataka. U daljnjem tekstu neće se objašnjavati relacijski model baze podataka već će biti opisan samo model podataka predstavljen putem razreda u programskom jeziku *Java*. Glavni razlog tome je što korištenje biblioteke *Hibernate* omogućava neovisnost implementiranog modela podataka u odnosu na bazu podataka jer se relacijski model stvara na temelju ostvarenog modela podataka te mu se pristupa prvenstveno koristeći sučelja iz biblioteke *Hibernate*. Preslikavanje se temelji na korištenju anotacija definiranih u programskom sučelju *Java Persistence* [10].

Sustavi međusobno komuniciraju preko binarnog konektora koji koristi unaprijed određeno TCP sučelje. Sustav *Ferko* prilikom slanja zahtjeva mora navesti indeks naredbe koja će se izvršiti na sustavu *Studtest2,* a struktura zahtjeva i odgovora točno je definirana za svaku naredbu.

U nastavku poglavlja prvo će biti opisane osnovne komponente na kojima se temelji provedbe ispitivanja u sustavu *Studtest2*. Potom će biti predstavljen model podataka za hijerarhiju nastavnih cjelina te sučelje za administraciju hijerarhije segmenata znanja implementiranu u sustavu *Ferko*. Na kraju poglavlja će biti opisana implementacija algoritma za adaptivno ispitivanje u sustavu *Studtest2*.

5.1 Koncepti sustava StudTest2

Sustav *Studtest2* pruža mogućnost izrade komponenata koji konceptualno predstavljaju zadatke. Navedene komponente se nazivaju *prleti* i mogu se stvarati korištenjem razvojnog okvira *Problem Development Framework* koji predstavlja jedan od glavnih dijelova sustava *Studtest2*. U nastavku će ukratko biti pojašnjeni koncepti na kojima se temelji struktura *prleta* i njegovo korištenje u svrhu provedbe e-ispitivanja.

Prlet se sastoji od 4 koncepta koji pokrivaju četiri zasebne funkcionalnosti :

- *ProblemGenerator* predstavlja obrazac za stvaranje zadataka te iz toga razloga sadržava sve potrebne informacija poput jedinstvenog identifikatora kojim je određen zadatak i popisa ostalih komponenti koje su zadužene za različite funkcionalnosti vezane uz taj zadatak;
- *ProblemEditor* predstavlja komponentu koja pruža mogućnost definiranja različitih parametara nužnih prilikom stvaranja zadatka i koji su propisani u komponenti *ProblemGenerator* tog zadatka;
- *ProblemInstantiator* predstavlja komponentu koja se koristi za stvaranje konkretnog zadatka pomoću *ProblemGeneratora* i parametara zadanih kroz *ProblemEditor*;
- *ProblemEvaluator* predstavlja komponentu za vrednovanje točnosti rješenja nakon što je ispitanik ponudio svoj odgovor na to konkretno pitanje.

Koncept *Problem* nastaje korištenjem *ProblemEditora* pri čemu zadani parametri služe za specificiranje ponašanja *ProblemGeneratora* čime se točno određuje što *Problem* ispituje. Primjerice, za predmet Digitalna logika moguće je ostvariti *prlet* koji bi ispitivao poznavanje algebarskog zapisa minterma ili maksterma nasumično odabrane funkcije. Pomoću stvorenog *ProblemEditora* određivao bi se broj parametara funkcije te bi bilo moguće odabrati da li taj zadatak traži upisivanje minterma ili maksterma.

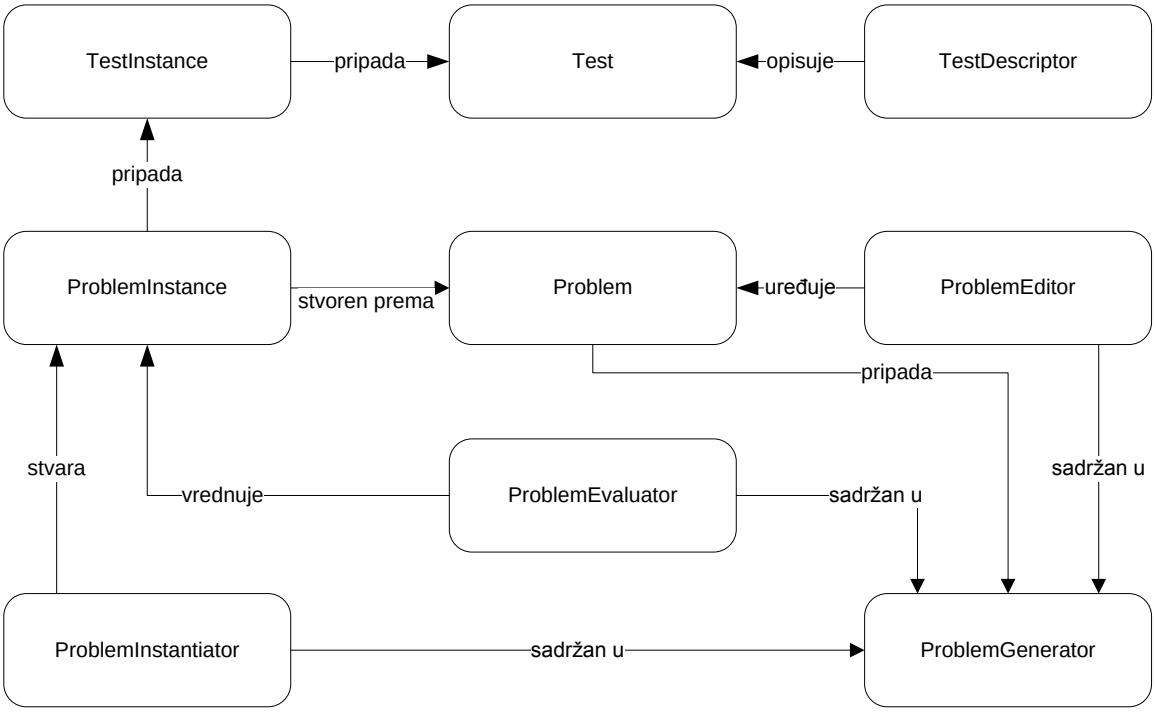
Koncept *TestDescriptor* predstavlja jedan opisnik ispita znanja i sadržava parametre potrebne za stvaranje jednog konkretnog ispita. Na primjer, navedeni

koncept može sadržavati podatke o vremenskom ograničenju, maksimalnom i minimalnom broju pitanja, sigurnosnim mehanizmima koji će biti korišteni tijekom ispitivanja i slično.

Koncept *TestInstance* predstavlja jedan konkretan ispit znanja koji je stvoren na temelju odgovarajućeg *TestDescriptora*. Instance koncepta *TestInstance* koje su definirane pomoću istog *TestDescriptora* pripadaju istom konceptu koji se naziva *Test*. Navedene pojmove najjednostavnije je objasniti pomoću primjera. Nastavnik na predmetu Digitalna logika definira novi ispit znanja pod nazivom „Prva domaća zadaća“ kojoj pristupaju svi studenti na kolegiju. To čini stvaranjem nove instance koncepta *TestDescriptor* i upisivanjem određenih parametara koji su nužni za provedbu tog ispita. Svakom studentu se dodjeljuje novostvoreni primjerak koncepta *TestInstance* prilikom prvog pristupa tom ispitu. Instanca koncepta *Test* koja je povezana uz prethodno stvoreni opisnik ispita sadrži listu svih primjeraka ispita odnosno u ovom slučaju sve primjerke ispita pod nazivom „Prva domaća zadaća“.

Prilikom definiranja novog ispita određuje se i fond pitanja iz kojeg će se moći birati pitanja za ispit. Točnije, potrebno je navesti koje instance koncepta *Problem* je moguće koristiti u provjeri znanja. Prilikom pokretanja ispita ili tijekom njegove provedbe se određuje koji će točno zadaci biti postavljeni ispitaniku te se za svaki zadatak stvaraju instance zadatka. Instanca zadatka modelirana je konceptom *ProblemInstance* i stvara se korištenjem odgovarajuće *ProblemInstantiatora* na temelju *Problema* koji odgovara tom zadatku. Za prethodno objašnjeni primjer zadatka u kojem je potrebno upisati odgovarajući algebarski zapis funkcije, stvara se novi primjerak koncepta *ProblemInstance* koji odgovara nekoj konkretnoj nasumičnoj odabranoj funkciji koja zadovoljava konfiguraciju zadanu korištenjem odgovarajućeg *ProblemEditora*.

Na slici 5-2 grafički su prikazani odnosi između prethodno opisanih koncepata koji su implementirani u sustavu *Studtest2*.



Slika 5-2. Odnos između dijela koncepata implementiranih u sustavu Studtest2

5.2 Model podataka za hijerarhiju nastavnih cjelina

U sustavu *StudTest2* definiran je model podataka za hijerarhijsku strukturu nastavnih cjelina i sadržan je u nekoliko razreda koji odgovaraju pojmovima objašnjenima u poglavlju 4. Taj model je samo manji dio složenog modela na kojem se temelji sustav za e-ispitivanje. Implementacija adaptivnog ispitivanja u sustavu *StudTest2* se obavljala u sklopu *TeachMe* projekta. Iz toga razloga velika većina razreda stvorenih u tom projektu u nazivu ima prefiks *TM*, kao npr. razred *TMSegment*.

Razred *TMSegment* predstavlja segment znanja u hijerarhijskoj strukturi nastavnih cjelina. Elementi razreda *TMSegment* predstavljeni su u tabeli 5-1.

Tablica 5-1. Važni elementi razreda TMSegment

Tip elementa	Naziv elementa	Opis elementa
String	URI	jedinstveni identifikator segmenta
String	title	naziv segmenta
double	difficulty	težina segmenta
integer	relevancy	relativna važnost segmenta u temi
TMTopic	parent	roditelj segmenta
Set<TMPrerequisite>	prerequisites	skup preduvjeta

Svaki segment određen je jedinstvenim identifikatorom segmenta odnosno URI-jem predstavljenim pomoću znakovnog niza. Uz jedinstveni identikator, potrebno je još definirati 4 parametra: naziv segmenta, težinu segmenta, relativnu važnost segmenta u temi te roditelja segmenta.

Težina segmenta je vrijednost s pomičnim zarezom na intervalu [0.0, 1.0] pri čemu donja granica odgovara segmentu koji uopće nije zahtjevan, a gornja granica jako zahtjevnom segmentu.

Relativna važnost segmenta u temi predstavlja glavni oslonac za određivanje ukupne važnosti pojedinog segmenta u stablu segmenata, a zadaje se u obliku cjelobrojne vrijednosti. Ako je neki segment dvostruko važniji od svih ostalih segmenata tada će i njegova mjera važnosti biti dvostruko veća od mjera ostalih segmenata. U tom slučaju kao njegovu mjeru možemo uzeti neku proizvoljnu pozitivnu cjelobrojnu vrijednost (primjerice 20), a za važnost svakog preostalog segmenata u temi postaviti dvostruko manju vrijednost (za dani primjer bi ona iznosila 10).

Nužno je pojasniti zašto su potrebne mjere težine i važnosti segmenta u ovom modelu jer bi se netko mogao zapitati o isplativosti definiranja važnosti segmenta i mogućnosti korištenja samo mjere težine u opisu elemenata hijerarhije nastavnih cjelina. Potreba za korištenjem obje mjere može se jednostavno objasniti postojanjem segmenata znanja koji nisu zahtjevni za savladavanje odnosno nemaju veliku težinu, ali su jako bitni za razumijevanje šireg konteksta na razini kolegija pa imaju i veliku važnost. Protuprimjer su segmenti koji su izrazito teški, a istovremeno ih nije potrebno u potpunosti razumjeti da bi uspješno svladali kolegij. Primjerice, to mogu biti matematički dokazi istog teorema pri čemu je moguće da su pojedini dokazi izrazito teški, a nije ih potrebno u potpunosti razumjeti kako bi se teorem koristio u nekim drugim segmentima znanja.

Svi segmenti su svrstani u teme odnosno svakom segmentu je dodijeljen segment koji ima ulogu roditelja. Jedina iznimka je korijen stabla segmenata koji predstavlja cjelokupan kolegij i iz toga razloga nema ni dodijeljenog roditelja.

Razredi *TMTopic* i *TMConcept* nasljeđuju apstraktan razred *TMSegment* te predstavljaju konkretne elemente hijerarhije nastavnih cjelina, teme odnosno koncepte. Elementi razreda *TMTopic* predstavljeni su u tablici 5-2.

Tablica 5-2. Elementi razreda TMTopic

Tip elementa	Naziv elementa	Opis elementa
Set<TMSegment>	segments	skup segmenata u temi

Svaki segment može imati skup preduvjeta koji su modelirani razredom *TMPrerequisite* (tablica 5-3). Razred *TMPrerequisite* sadrži referencu na segment kojem pripada taj preduvjet odnosno na vlasnika preduvjeta, ali i referencu na segment koji je preduvjet vlasniku. Razina poznavanja preduvjeta se pohranjuje kao vrijednost s pomičnim zarezom na intervalu [0.0, 1.0] gdje donja granica predstavlja situaciju u kojoj uopće nije potrebno imati znanje o segmentu koji je preduvjet, a gornja granica situaciju u kojoj u potpunosti treba razumjeti preduvjet kako bi postojala mogućnost uspješnog savladavanja tog određenog segmenta.

Tablica 5-3. Važni elementi razreda TMPrerequisite

Tip elementa	Naziv elementa	Opis elementa
TMSegment	owner	segment koji je vlasnik preduvjeta
TMSegment	prerequisiteSegment	segment koji je preduvjet
double	level	razina poznavanja preduvjeta

Razred *Workspace* predstavlja radnu grupu odnosno kolegij čiji je naziv jednak nazivu kolegija u sustavu *Ferko*. Razred *TMSegmentTree* (tablica 5-4) koristi se za provjeru verzije stabla te sadrži reference na odgovarajuće primjerke razreda *Workspace* i razreda *TMSegment* koji predstavlja korijen stabla segmenata. Verzija stabla segmenata se inkrementalno povećava prilikom svake promjene stabla.

Tablica 5-4. Važni elementi razreda TMSegmentTree

Tip elementa	Naziv elementa	Opis elementa
TMSegment	rootSegment	korijen stabla segmenata
Workspace	workspace	radna grupa odnosno kolegij
long	version	verzija stabla segmenata

5.3 Sučelje za administraciju hijerarhije segmenata znanja

Sučelje za administraciju hijerarhije segmenata znanja (u daljnjem tekstu *TeachMeAdmin*) je implementirano u sustavu *Ferko* koristeći tehnologiju *Java Applet* [11]. *TeachMeAdmin* manipulira modelom podataka implementiranim u sustavu *StudTest2* posredovanjem sustava *Ferko*. Razlog tome je ograničenje u tehnologiji *Java Applet* koje dozvoljava komunikaciju samo s web-poslužiteljem s kojeg je dohvaćen *aplet* dok je komunikacija s ostalim poslužiteljima zabranjena.

Za komunikaciju između *apleta* i sustava *StudTest2* definirana je struktura zahtjeva i odgovora u XML formatu. Na slici 5-3 je prikazan zahtjev za dohvatom segmenta koji se šalje sustavu *Studtest2*, a prikazan je i odgovor kojeg *StudTest2* šalje *apletu* uz posredovanje sustava *Ferko*.

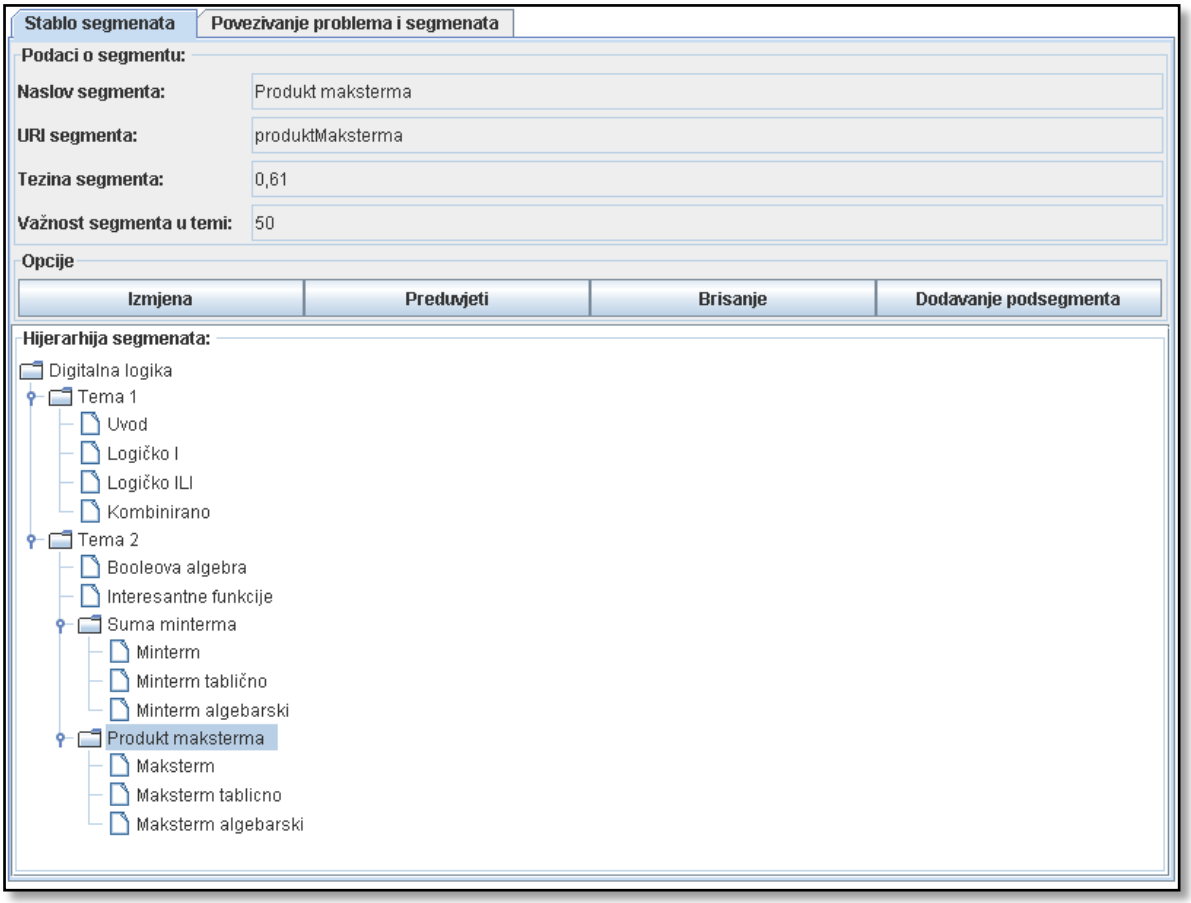
```
[ZAHTJEV]:
<?xml version="1.0" encoding="UTF-8"?>
<request>
    <operation>getSegment</operation>
    <data>
        <segmentTree version="151" /><uri>http://studtest2.zemris.fer.hr/19674/sumaMinterma</uri>
        <numberOfLevels>2</numberOfLevels>
    </data>
</request>

[ODGOVOR]:
<?xml version="1.0" encoding="UTF-8"?>
<response>
    <result successful="true" />
    <data>
        <segmentTree version="151" />
        <segment type="topic" uri="http://studtest2.zemris.fer.hr/19674/sumaMinterma"
                 title="Suma minterma" difficulty="0.51" relevancy="50" >
            <parent uri="http://studtest2.zemris.fer.hr/19674/tema2" />
            <children size="3" >
                <segment type="concept" uri="http://studtest2.zemris.fer.hr/19674/mintermTablicno"
                         title="Minterm tablično" difficulty="0.51" relevancy="50" ></segment>
                <segment type="concept" uri="http://studtest2.zemris.fer.hr/19674/mintermAlgebarski"
                         title="Minterm algebarski" difficulty="0.51" relevancy="50" ></segment>
                <segment type="concept" uri="http://studtest2.zemris.fer.hr/19674/minterm"
                         title="Minterm" difficulty="0.51" relevancy="50" ></segment>
            </children>
        </segment>
    </data>
</response>
```

Slika 5-3. Prikaz zahtjeva i odgovora u komunikaciji između appleta i sustava StudTest2

Pristup sučelju za administraciju hijerarhije segmenata znanja imaju svi korisnici sustava koji se smatraju nastavnim osobljem na kolegiju. Nastavno osoblje prije korištenja sučelja mora osigurati postojanje odgovarajuće instance koncepta *Workspace* u sustavu *StudTest2* za taj kolegij. Ako u sustavu *StudTest2* ne postoji instanca koncepta *Workspace* koja ima jednako ime kao i kolegij na *Ferku*, neće biti dozvoljeno učitavanje *apleta*.

Aplet prilikom pokretanja šalje zahtjev za dohvat korijena stabla segmenata za taj kolegij. Sustav *StudTest2* provjerava je li već postoji korijen stabla vezan uz navedenu radnu grupu te ga stvara u slučaju da on ne postoji.

Korištenje sučelja će biti pojašnjeno na primjeru predmeta Digitalna logika za kojeg je stvoreno stablo segmenata znanja. Prikazano stablo zbog jednostavnosti pokriva samo manji dio tog kolegija.

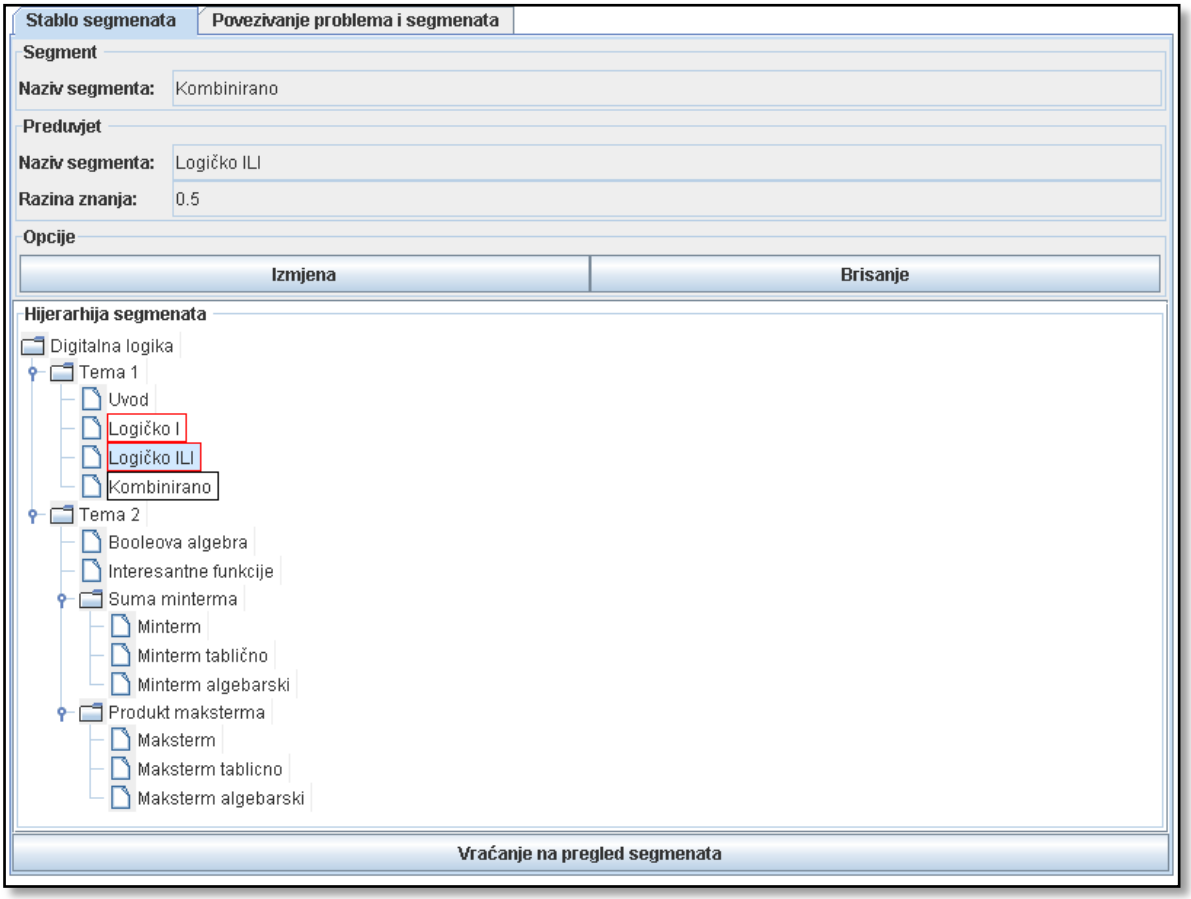


Slika 5-4. Pregled stabla segmenata znanja

Sučelje za administraciju hijerarhije segmenata znanja dijeli se na dva glavna dijela:

- pregled stabla segmenta i
- pregled povezivanja zadataka odnosno problema sa odgovarajućim segmentima.

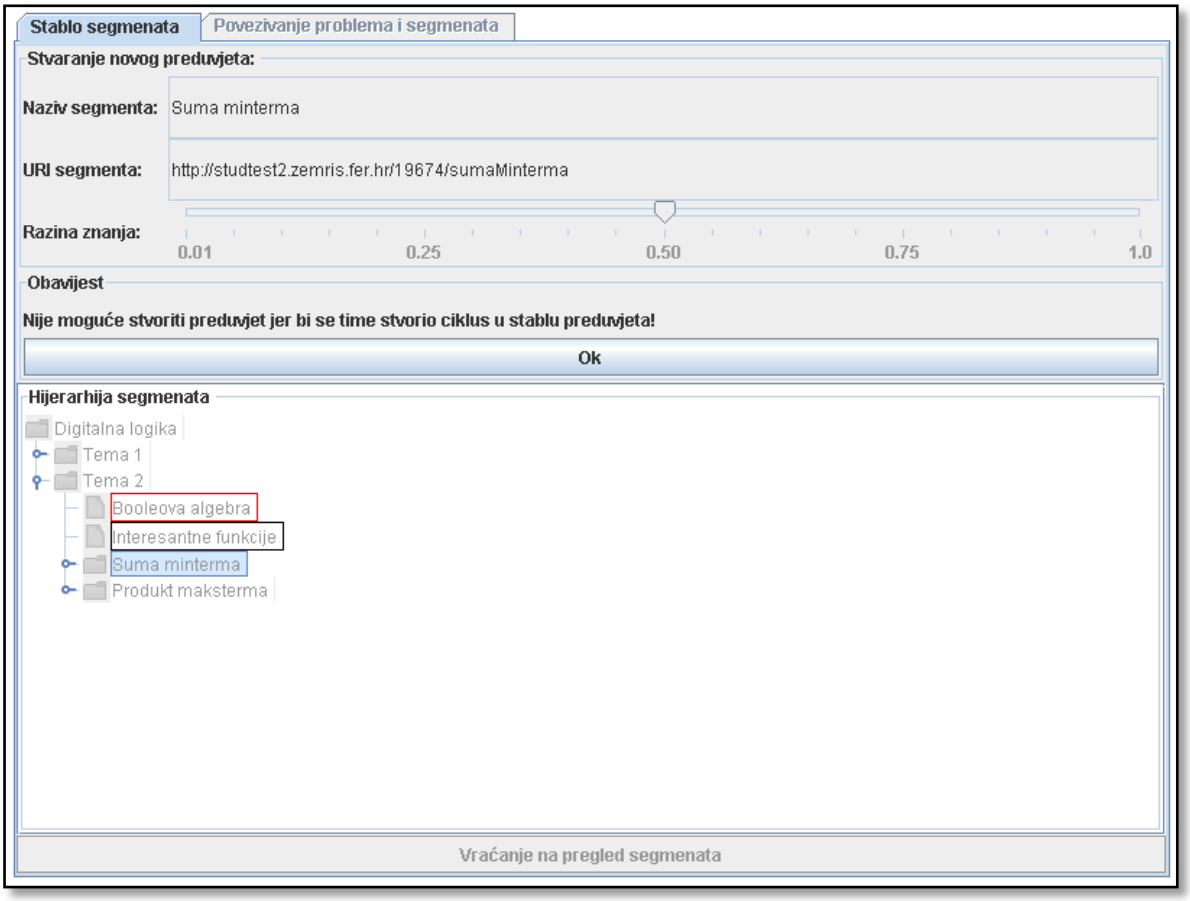
Pregled stabla segmenata (slika 5-4) pruža mogućnosti za izmjenu i brisanje postojećih te stvaranje novih segmenata uz mogućnost unošenja svih potrebnih parametara opisanih u poglavlju 5.2. Za svaki segment je moguće pregledavati, mijenjati i brisati postojeće te stvarati nove preduvjete. Slika 5-5 predstavlja prikaz preduvjeta za segment pod nazivom „Kombinirano“ gdje vidimo da su za preduvjete odabrani segmenti „Logičko I“ i „Logičko ILI“.



Slika 5-5. Pregled preduvjeta za odabrani segment

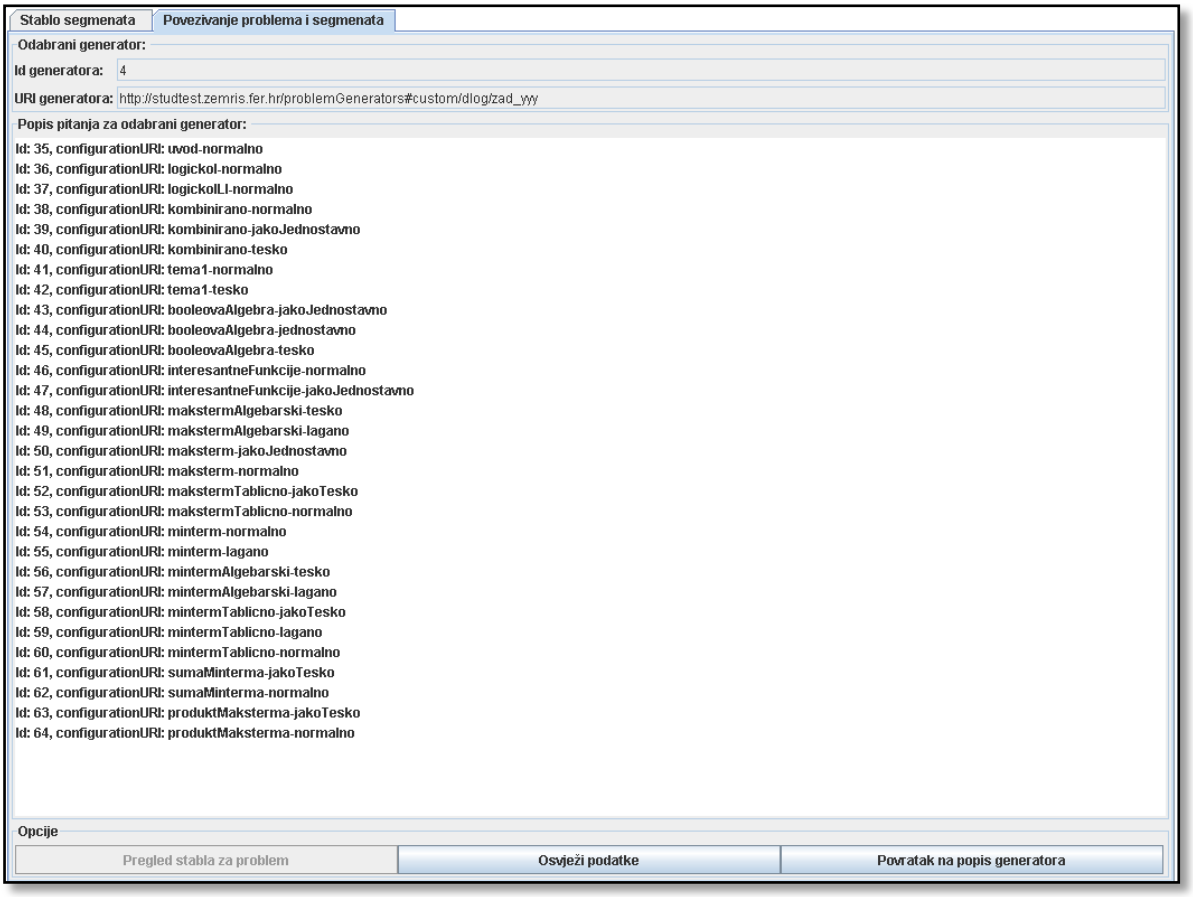
U poglavlju 5.2 je definirano da graf preduvjeta ne može sadržavati ciklus te je bilo potrebno onemogućiti dodavanje novog preduvjeta koristeći *TeachMeAdmin* kojim bi se stvorio ciklus u grafu preduvjeta. Ciklus preduvjeta može biti sastavljen i preduvjeta nekog od predaka čvora u stablu segmenata. Primjerice, ako je prethodno definirano da je koncept *Interesantne funkcije* preduvjet temi *Suma minterma* tada se smatra kako je koncept *Interesantne funkcije* ujedno i indirektni preduvjet svim konceptima toj temi. U tom bi se slučaju postupkom dodavanja koncepta *Minterm* kao preduvjeta segmenta *Interesantne funkcije* stvorio ciklus preduvjeta što se nikako ne smije dozvoliti.

Na slici 5-6 prikazan je pokušaj dodavanja teme *Suma minterma* kao preduvjeta koncepta *Interesantne funkcije* što *TeachMeAdmin* ne dozvoljava iz razloga što je prethodno definirano da *Suma minterma* za preduvjet ima *Interesantne funkcije*.



Slika 5-6. Primjer pokušava dodavanja preduvjeta kojim bi se stvorio ciklus u grafu preduvjeta

Druga važna funkcionalnost implementirana u sučelju *TeachMeAdmin* je povezivanje problema i segmenata znanja koje taj problem ispituje. Korisniku je dan popis svih generatora problema vezanih uz odgovarajuću radnu grupu odnosno instancu razreda *Workspace*. Nakon odabira nekog generatora prikazuju se sve instance razreda *Problem* povezane s tim generatorom (slika 5-7).

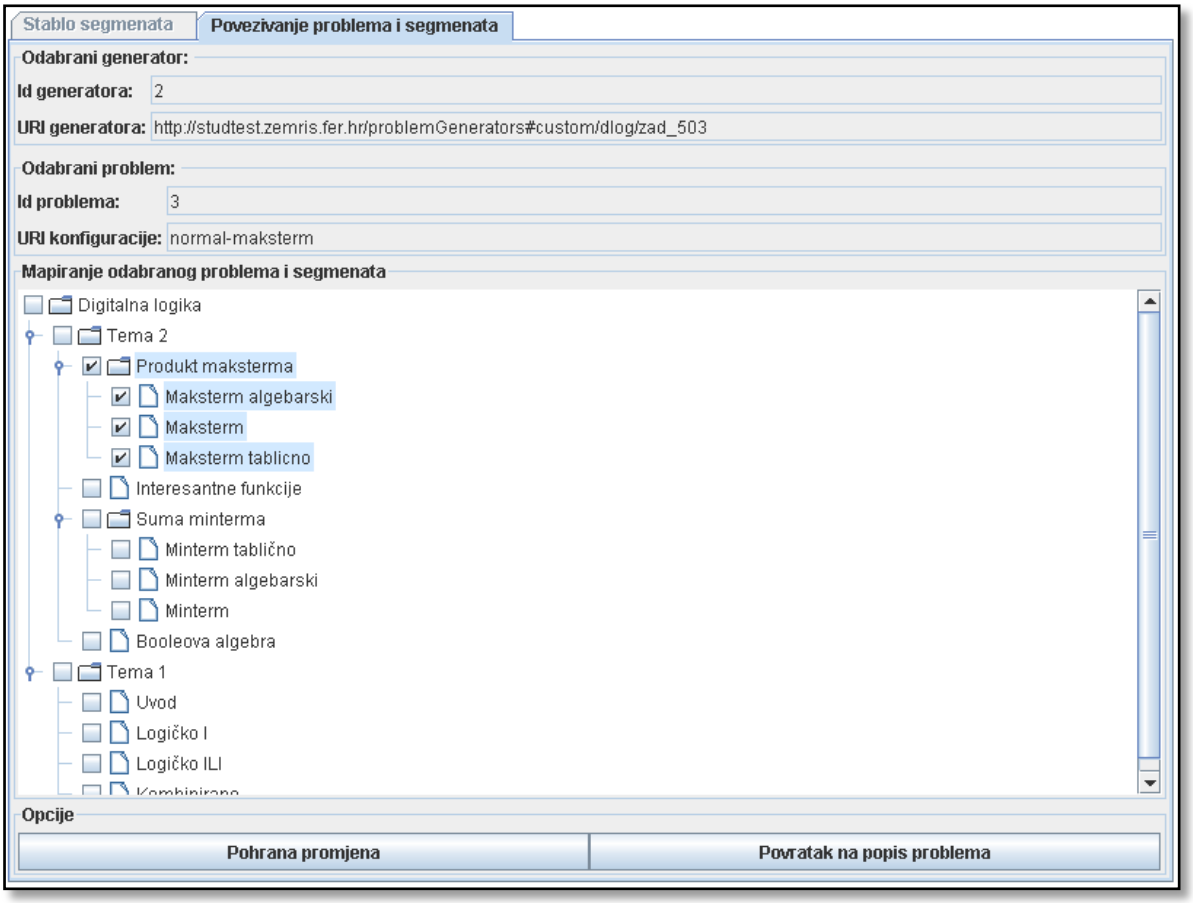


5-7. Primjer popisa problema za izabrani generator problema

Sučelje *TeachMeAdmin* za svaki problem omogućava unošenje potrebnih parametara težine pitanja, prosječnog trajanja u sekundama te označavanja koje teme i koncepte u stablu segmenata znanja pokriva taj problem (slika 5-8).

Težina pitanja može biti odabrana između sljedećih ponuđenih vrijednosti:

- jako teško pitanje,
- teško pitanje,
- pitanje normalne težine,
- jednostavno pitanje i
- jako jednostavno pitanje.



Slika 5-8. Primjer definiranja parametara za neki problem

5.4 Implementacija algoritma za adaptivno ispitivanje

U sustavu StudTest2 su definirana sučelja za komponente koje se koriste u provedbi ispita. Komponenta koja implementira algoritam za ispitivanje mora implementirati sve metode koje su propisane u sučelju *ITestController*. U tablici 5-5 su navedena imena i opisi važnih metoda koje propisuje to sučelje (parametri metoda i tip povratne vrijednosti ispušteni su zbog preglednosti).

Tablica 5-5. Važne metode sučelja ITestController

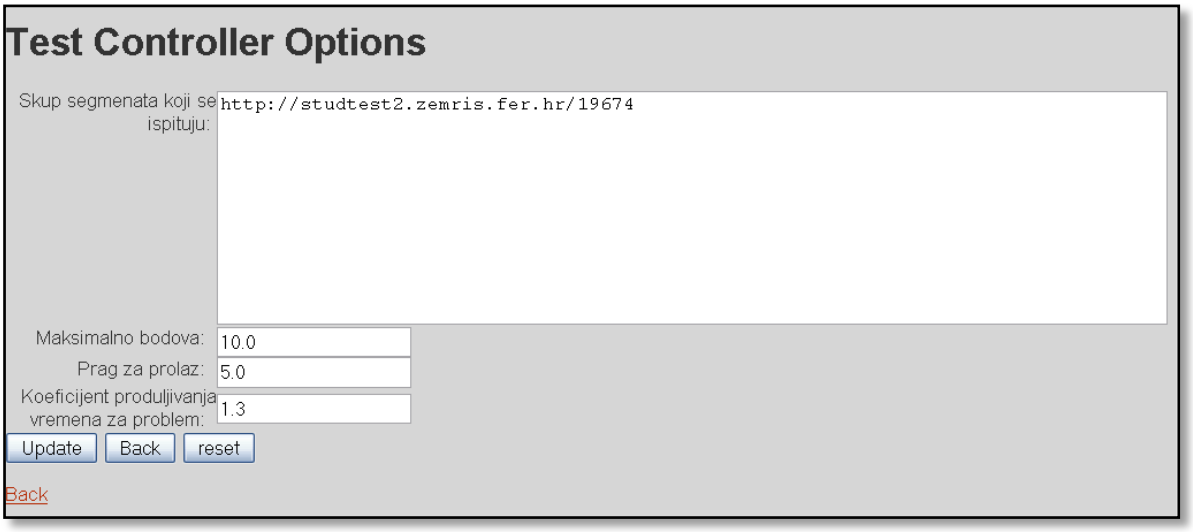
Naziv metode	Opis metode
`isAdaptiveAssessment`	Metoda vraća pozitivnu logičku vrijednost ako se ispitivanje temelji na algoritmu za adaptivno ispitivanje, u suprotnom je ta vrijednost negativna.
`onAction`	Metoda obrađuje odabranu akciju od strane ispitanika te određuje hoće li se ispitivanje nastaviti, prekinuti ili zamrznuti.
`getTaskRangeToDisplay`	Metoda utvrđuje koji će se skup zadataka prikazati ispitaniku.
`getActions`	Metoda dohvaća listu akcija koja se prikazuju ispitaniku i koje ispitanik može koristiti.
`prepareTestInstance`	Metoda koja se poziva prilikom pokretanja ispita i u kojoj se obavljaju sve pripreme za uspješno provođenje ispita.
`calculateAndSetScore`	Metoda računa konačno bodovno stanje ispita te određuje je li ispitanik položio ispit ili ga nije položio.

Za pohranu privatnih podataka svake komponente koristi se koncept repozitorija podataka. Primjerice, u repozitorije se spremaju svi parametri koje ispitivač unosi za odabrane komponente kroz *StudTest-webadmin* sučelje.

Za implementaciju algoritma za adaptivno ispitivanje u sustavu StudTest2 je stvoren razred *TeachMeTestController* koji sadrži sve metode propisane sučeljem *ITestController*. Objašnjenje algoritma se biti razloženo u opisu postupaka koji se

izvršavaju u metodama `prepareTestInstance`, `onAction` i `calculateAndSetScore`.

Prvi korak u izvedbi ispita je konfiguriranje instance *TestDescriptora* i svih ostalih odabranih komponenti. Konfiguracija upravljača provjere znanja za adaptivno ispitivanje je prikazana na slici 5-9.



Slika 5-9. Parametri za konfiguriranje komponente TeachMeTestController

Nužni parametri za provedbu implementiranog adaptivnog ispitivanja su:

- *popis segmenata koji se ispituju* – potrebno je upisati samo korijene podstabala koje želimo ispitati; primjerice, na slici 5-9 naveden je URI segmenta koji predstavlja kolegij *Digitalna logika* što znači da će se ispitivati svi segmenti koji se nalaze u tom stablu segmenata;
- *maksimalan broj bodova* – najveći broj bodova koji se može dobiti u slučaju da je ispit riješen u potpunosti točno;
- *prag za prolaz* – minimalni broj bodova koji ispitanik mora imati na kraju ispita kako bi se smatralo da je ispitanik uspješno riješio test;
- *koeficijent produljivanja vremena za problem* – navedeni parametar se koristi prilikom određivanja maksimalnog vremena za rješavanje nekog zadatka T_{max} i to po formuli $T_{max} = T_{avg} * E_T$ gdje je T_{avg}

prosječno vrijeme rješavanja određeno za svaki problem, a E_T koeficijent produljivanja vremena.

Priprema za izvedbu ispita se obavlja u metodi `prepareTestInstance` u kojoj se dohvaćaju svi korijeni podstabala navedeni prilikom definiranja ispita. Provjerava se ispravnost toga unosa jer ispitivač može upisati i URI-je segmenata koji se nalaze u podstablu čiji je korijen neki već navedeni segment. To se jednostavnije može pojasniti na temelju primjera sa slike 5-4. Ako ispitivač u konfiguraciji upiše da se ispituju segmenti *Suma minterma* i *Minterm*, u ovoj metodi će se filtrirati ti segmenti te će se kao korijen podstabla prepoznati samo *Suma minterma* jer ta tema sadrži koncept *Minterm*.

Nakon tog postupka se obavlja određivanje glavnog podstabla cijelog stabla segmenata koji sadrži sve segmente koji se ispituju i koji ima najmanje čvorova od svih mogućih podstabala. Ako se za stablo segmenata sa slike 5-4 definira da se ispituju segmenti *Interesantne funkcije* i *Suma minterma*, tada se kao glavno podstablo uzima ono čiji je korijen segment *Tema2*.

Tijekom ispitivanja je nužno imati podatke o svakom segmentu koji se ispituje. To se moglo ostvariti na dva načina; spremanjem informacija direktno u bazu podataka ili indirektno koristeći repozitorij podataka instance testa. Zbog jednostavnosti je korištena druga metoda, a time je osigurano da eventualne promjene u stablu segmenata od strane administratora kolegija ne utječu na ispite koji su u tijeku. Implementiran je postupak pohrane informacija o prethodno određenom glavnom podstablu koje sadrži sve segmente i to u XML formatu. Prilikom pohrane ignoriraju se sva podstabla u kojima nema segmenata koji se ispituju, a ignoriraju se i svi preduvjeti u kojima je definiran neki od segmenata koji se ne ispituje.

Određuje se uređeni poredak segmenata u kojem nije moguće da postoji segment koji se nalazi u nizu prije bilo kojeg segmenta koji mu je izravni ili implicitni preduvjet ili mu je roditelj u stablu segmenata. To je moguće ostvariti za svako stablo segmenata koje je izgrađeno uz poštivanje pravila definiranih u poglavlju 4. jer se u tom slučaju graf preduvjeta aciklički usmjereni graf od kojeg se uvijek može formirati definirani uređeni poredak.

U metodi `onAction` se obrađuje akcija koju je odabrao ispitanik. Metoda mijenja procjenu znanja ispitanika nakon odgovora na pitanje te na temelju te procjene bira iduće pitanje. Odabir idućeg pitanja može se podijeliti na nekoliko manjih koraka:

1. izračunava se vrijeme T_b po formuli $T_b = T_{rest} / E_t$ gdje T_{rest} predstavlja preostalo vrijeme za rješavanje ispita, a E_t koeficijent produljivanja vremena potrebnog za rješavanje problema;

2. dohvaćanje liste problema koji još nisu iskorišteni u ispitu pri čemu se ignoriraju problemi za koje nije definirano prosječno vrijeme rješavanja i težina te čije prosječno vrijeme nije veće od T_b;

3. provjerava se broj problema u listi te ako postoji samo jedan problem, ispitaniku se postavlja to pitanje i nudi samo akcija za završavanje ispita;

4. ako lista sadrži više od jednog problema, za svaki problem računa se zbroj važnosti segmenata u stablu koje problem pokriva te se taj zbroj dijeli sa brojem segmenata koje pitanje pokriva kako bi se dobila mjera S_i koja je prosječna važnost po broju segmenata;

5. izračunava se srednje prosječno vrijeme $T_{avgList}$ za listu problema na način da se zbroje vremena T_{avg} za sva pitanja u listi te se dobivena suma podijeli s brojem pitanja;

6. za svako pitanje se izračunava odstupanje T_{var} od srednjeg prosječnog vremena pri čemu se posebno zabilježi maksimalno odstupanje T_{varMax};

7. za svaku pitanje se računa vrijednost C_i na temelju formule
$$C_i = S_i * (2 - T_{var} / T_{varMax})$$

8. odabir idućeg pitanja se vrši korištenjem jednostavne selekcije proporcionalne izračunatim vrijednostima C_i (engl. *roulette-wheel method*).

Važnost segmenata R_i u stablu računa se na temelju rekurzivne metode koja se izvršava od korijena prema listovima stabla. Korijenu stabla dodjeljuje se $R_i = 1$, a za ostale segmente se računa kroz korake:

1. za svaki segment u temi izračunati važnost R_i koja se izračunava kao umnožak postotka trenutne važnosti segmenta u toj temi i važnosti R_p koja predstavlja važnost teme u kojoj se nalazi segment;

2. ako je segment tema tada je potrebno obraditi sve segmente u toj temi prema postupku definiranom u 1. koraku.

Prosječna važnost po broju segmenata S_i koja se računa za svaki problem je bitna zbog namjere da problemi koji pokrivaju manji broj vrlo važnih segmenata imaju veću vjerojatnost odabira od problema koji pokrivaju puno segmenata manje važnosti.

Procjena znanja studenta o elementima iz stabla segmenata se mijenja nakon svakog postavljenog pitanja i to u ovisnosti o točnosti ispitanikovog odgovora na pitanje. Također, mijenja se i trenutna procjena važnosti segment u temi jer se na temelju te važnosti mijenja vjerojatnost odabira pitanja. Time se algoritam usmjerava ka ispitivanju dijelova stabla segmenata koji nisu dovoljno ispitani. Početna vrijednost trenutne važnosti segmenta je jednaka originalnoj važnosti segmenta koju je upisao administrator kolegija kroz sučelje *TeachMeAdmin*.

Prilikom prvog pozivanja metode `onAction` pronalaze se svi segmenti koji nemaju preduvjete te im se početne trenutne važnosti uvećavaju za koeficijent 10. Na taj način se izravno utječe na vjerojatnost odabira idućeg pitanja jer se time potiče na postavljanje pitanja koja pokrivaju segmente bez preduvjeta u ranoj fazi provedbe ispita. Segmenti bez preduvjeta su najčešće preduvjeti drugim segmentima što je izuzetno važno zbog načina na koji preduvjeti utječu na procjenu drugih segmenata u stablu segmenata znanja. Svi segmenti kojima je na taj način povećana trenutna važnost se dodaju u listu segmenata s povećanom važnosti.

U metodi `onAction` se dohvaća točnost zadnjeg postavljenog pitanja te se ta vrijednost pohranjuje u svaki segment kojeg to pitanje ispituje. Nakon toga se pristupa promjeni procjene znanja za sve segmente i promjeni trenutne važnosti segmenata u stablu.

Promjena procjene znanja za segmente se obavlja na način da se segmenti obrađuju po redu koji odgovara uređenom poretku stvorenom u metodi `prepareTestInstance`. Za svaki koncept se provode sljedeći postupci:

1. izračunava se mjera utjecaja preduvjeta M na temelju formule

$$M = \frac{\sum_i^n W_i \ * \ L_i * \ K_i}{\sum_i^n W_i \ * \ L_i}$$

   gdje je W_i težina segmenta koji je preduvjet P_i, K_i mjera procjene znanja segmenta koji je preduvjet P_i, a L_i razina poznavanja segmenta iz preduvjeta P_i koja je nužna za uspješno savladavanje segmenta;

2. izračunava se mjera A koja je definirana kao težinski prosjek točnosti odgovora na sva pitanja Z_i koja pokrivaju taj segment prema formuli

$$A = \frac{\sum_i^n D_i \ * \ V_i}{\sum_i^n D_i}$$

   gdje je D_i težina pitanja Z_i, a V_i točnost ispitanikovog odgovora na pitanje Z_i;

3. postoje četiri različita slučaja pri izračunu procjene znanja koncepta:

   a. ako koncept ima preduvjete i postoje problemi koji su ispitivali taj koncept, kao procjena se uzima umnožak $A * 0{,}7 + M * 0{,}3$,

   b. ako koncept nema preduvjete i postoje problemi koji su ispitivali taj koncept, kao procjena se uzima vrijednost A koja je izračunata za taj koncept,

   c. ako koncept ima preduvjete i ne postoji niti jedan problem koji je ispitivao taj koncept, kao procjena se uzima vrijednost M koja je izračunata za taj koncept,

d. ako koncept nema ni preduvjeta niti postoje problemi koji su ispitivali taj koncept, pretpostavlja se da ispitanik u potpunosti zna taj koncept pa se za kao vrijednost procjene uzima vrijednost 1,00.

Za svaku temu se provode sljedeći postupci:

1. izračunava se mjera utjecaja preduvjeta M na temelju formule

$$M = \frac{\sum_i^n W_i \ * \ L_i * \ K_i}{\sum_i^n W_i \ * \ L_i}$$

gdje je W_i težina segmenta koji je preduvjet P_i, K_i mjera procjene znanja segmenta koji je preduvjet P_i, a L_i razina poznavanja segmenta iz preduvjeta P_i koja je nužna za uspješno savladavanje segmenta;

2. izračunava se mjera E koja predstavlja ukupnu procjenu znanja segmenata u temi i definirana je formulom

$$E = \frac{\sum_i^n B_i * \ K_i}{\sum_i^n B_i}$$

gdje je B_i inicijalna važnost segmenta u temi odnosno važnost koju je segment imao na početku ispitivanja, a K_i procjena znanja segmenta;

3. izračunava se mjera Q koja predstavlja težinski prosjek točnosti odgovora na pitanja koja pokrivaju tu temu i definirana je formulom

$$Q = \frac{\sum_i^n D_i \ * \ V_i}{\sum_i^n D_i}$$

gdje je D_i težina pitanja Z_i, a V_i točnost ispitanikovog odgovora na pitanje Z_i;

4. postoji osam različitih slučajeva pri izračunu procjene znanja teme:

   a. ako postoji barem jedan riješeni problem koji pokriva tu temu i u temi se nalazi barem jedan segment te tema ima preduvjeta, procjena se računa na temelju formule $Q * 0{,}5 + E * 0{,}3 + M * 0{,}2$,

b. ako postoji barem jedan riješeni problem koji pokriva tu temu i u temi se nalazi barem jedan segment te tema nema preduvjeta, procjena se računa na temelju formule $Q * 0{,}6 + E * 0{,}4$,

c. ako se u temi nalazi barem jedan segment i tema ima preduvjeta te ne postoji niti jedan riješeni problem koji pokriva tu temu, procjena se računa na temelju formule $E * 0{,}7 + M * 0{,}3$,

d. ako tema ima preduvjeta i postoji barem jedan riješeni problem koji pokriva tu temu te se u temi ne nalazi niti jedan segment, procjena se računa na temelju formule $Q * 0{,}7 + M * 0{,}3$,

e. ako se u temi nalazi barem jedan segment i tema nema preduvjeta te ne postoji niti jedan riješeni problem koji pokriva tu temu, kao procjena se uzima vrijednost E izračunata za tu temu,

f. ako tema ima preduvjeta i u temi se ne nalazi niti jedan segment te ne postoji niti jedan riješeni problem koji pokriva tu temu, kao procjena se uzima vrijednost M izračunata za tu temu,

g. ako postoji barem jedan riješeni problem koji pokriva tu temu i tema nema preduvjeta te se u temi ne nalazi niti jedan segment, kao procjena se uzima vrijednost Q izračunata za tu temu ili

h. ako ne postoji niti jedan riješeni problem koji pokriva tu temu i tema nema preduvjeta te se u temi ne nalazi niti jedan segment, pretpostavlja se da ispitanik u potpunosti zna taj koncept pa se kao vrijednost procjene uzima vrijednost 1,00.

Promjena važnosti se također obavlja nakon odgovora na svako pitanje kroz korake:

1. dohvatiti zadnje pitanje na koje je ispitanik dao odgovor;
2. ako to pitanje pokriva neki segment koji se nalazi u listi segmenata s povećanom važnosti, maknuti taj segment iz liste te vratiti trenutnu važnost segmenta na vrijednost koju je segment imao prije povećanja;

3. trenutna važnost svih segmenata koji za preduvjete imaju nekog od segmenata koje ispituje zadnji prikazani problem se mijenja na način da se

    a. povećava za 20% ako je točnost odgovora na pitanje veća od 50% ili

    b. smanjuje za 20% ako je točnost odgovora na pitanje manja od 50%;

4. trenutna važnost segmenata koje pokriva zadnje pitanje se smanjuje u ovisnosti o težini pitanja i to na način da

    a. teža pitanja uzrokuju veće smanjenje trenutne važnosti segmenta nego što to uzrokuju jednostavnija pitanja ako je točnost odgovora na pitanje veća od 50% odnosno

    b. jednostavnija pitanja uzrokuju veće smanjenje trenutne važnosti nego što to uzrokuju teža pitanja ako je točnost odgovora na pitanje manja od 50%;

5. u listu segmenata s povećanom važnosti se dodaju segmenti čiji su svi preduvjeti već ispitani tijekom ispitivanja te im se povećava trenutna važnost za koeficijent 10; ako je neki od preduvjeta tema tada se segment dodaje u listu samo ako su ispitani svi segmenti u temi koja je preduvjet ili ako barem jedno od prethodno postavljenih pitanja pokriva tu temu.

U metodi `calculateAndSetScore` se računa konačno bodovno stanje na način da se odredi postotak procjene znanja svih segmenata u stablu te se dobivena vrijednost pomnoži s maksimalnim brojem bodova definiranim tijekom konfiguracije ispita. Postotak procjene znanja stabla E se računa putem formule

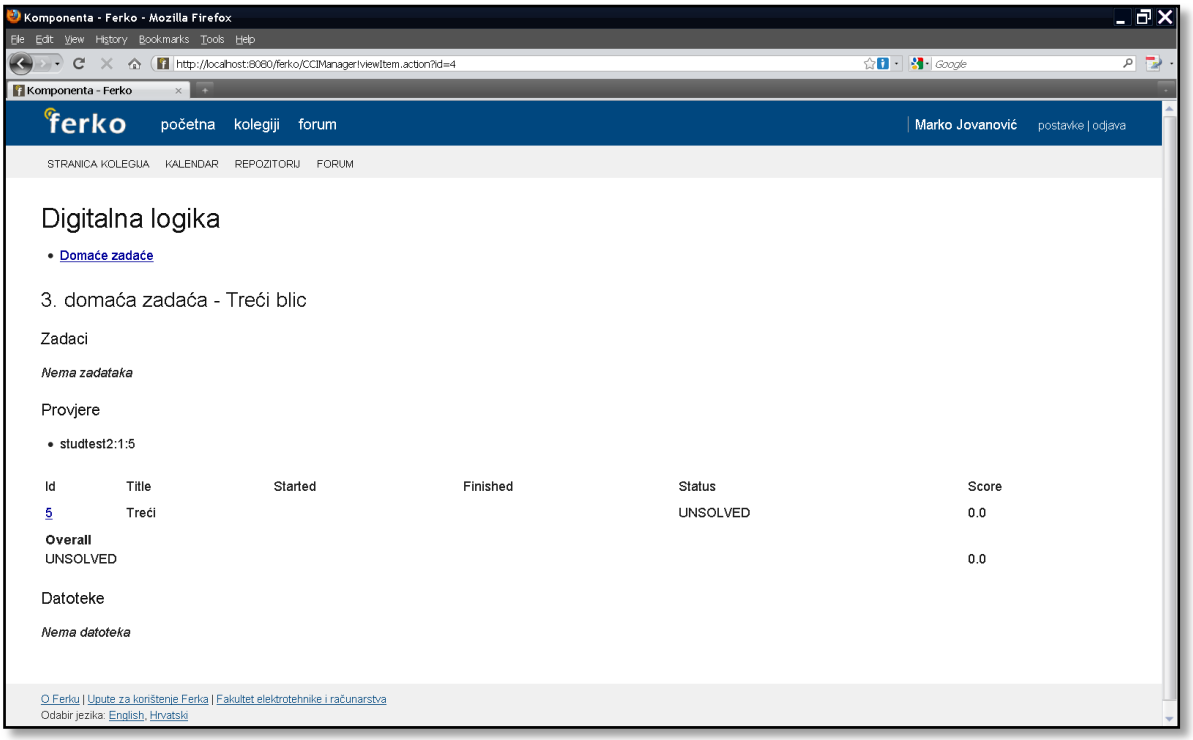
$$E = \frac{\sum_i^n R_i * K_i}{\sum_i^n R_i}$$

gdje R_i predstavlja važnost segmenta u stablu, a K_i predstavlja procjenu znanja segmenta pri čemu se za izračun koriste samo segmenti koji su korijeni podstabala određeni u metodi `prepareTestInstance`. Na kraju se provodi odluka o konačnom rezultatu ispitivanja. Ispitanik ima pozitivan rezultat ako je konačno bodovno stanje veće ili jednako definiranom pragu za prolaz koji se zadaje prilikom konfiguracije ispita dok u suprotnom nije položio ispit.

5.5 Primjer izvođenja adaptivnog ispitivanja u sustavu Ferko

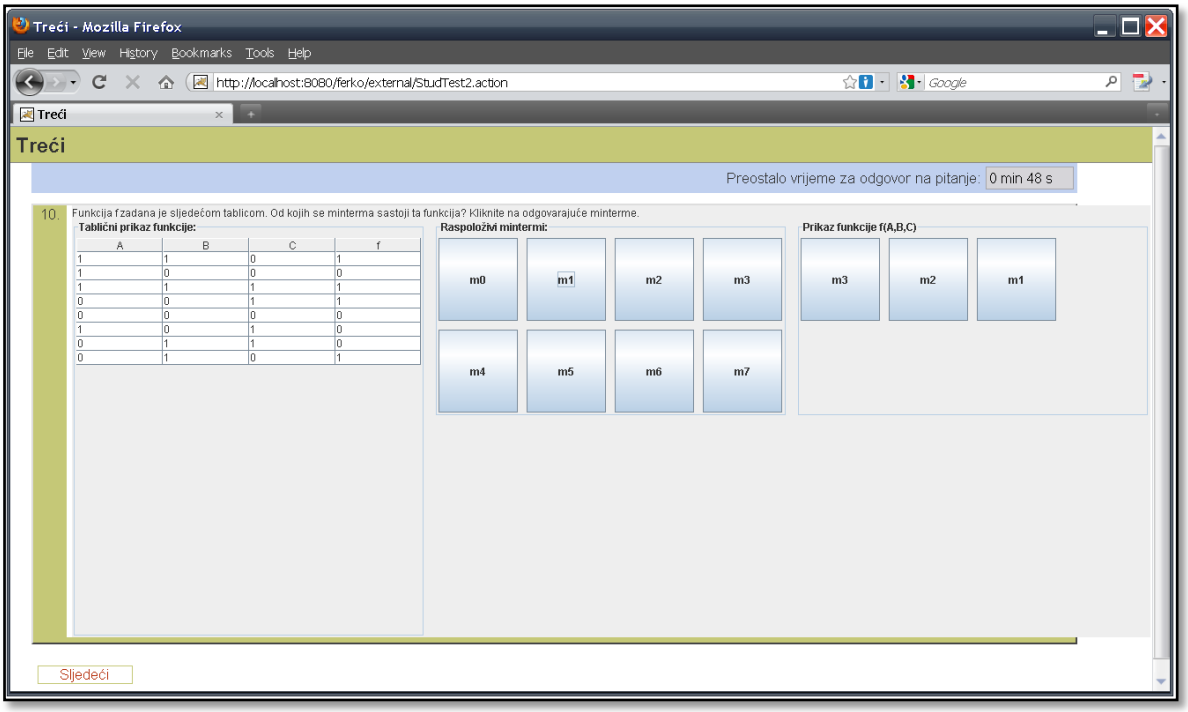
Studenti ispitu pristupaju kroz sustav *Ferko* nakon što je nastavno osoblje ispunilo sve potrebne preduvjete za provedbu ispitivanja. Za adaptivno ispitivanje je potrebno stvoriti hijerarhiju segmenata znanja te povezati svako pitanje iz fonda sa segmentima koje pokriva to pitanje što je već prethodno objašnjeno u ovom poglavlju.

Na slici 5-10 prikazan je primjer pristupanja rješavanju ispita na kolegiju Digitalna logika. Ispit se pokreće odabirom odgovarajućeg identifikatora ispita.

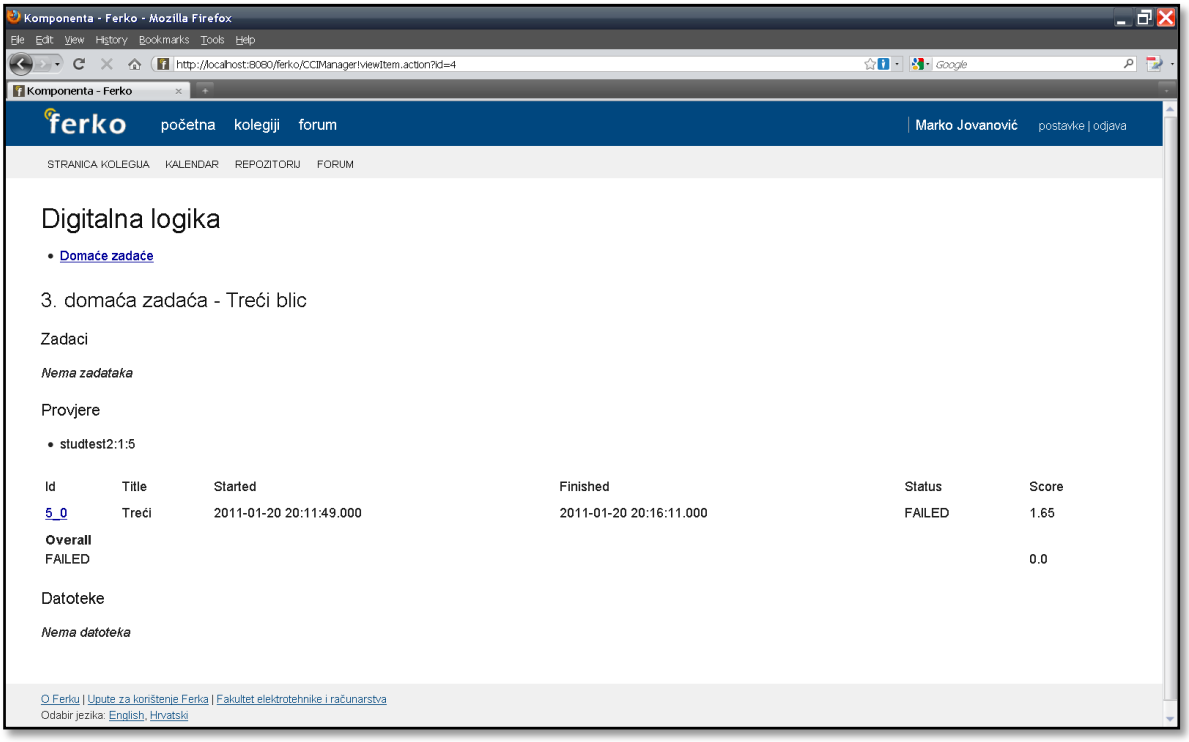


5-10. Primjer pristupanja rješavanju ispita

Na slici 5-11 prikazana je provedba ispitivanja na kolegiju Digitalna logika odnosno zadatak koji ispituje sumu minterma. U gornjem desnom uglu se nalazi preostalo vrijeme za odgovaranje na pitanje. Trenutno rješenje problema se automatski šalje na web-poslužitelj u trenutku kada istekne vrijeme za rješavanje zadatka te se ispitaniku prikazuje novo pitanje ili se završava ispit u slučaju da je zadovoljen određeni kriterij zaustavljanja. Na slici 5-12 prikazan je konačan rezultat ispitivanja. Vidljivo je da ispitanik nije uspješno riješio ispit jer je konačno bodovno stanje manje od praga za prolaz koji je definiran na slici 5-9.



5-11. Primjer rješavanja zadatka



5-12. Prikaz rezultata ispitivanja

6 Zaključak

Metode za adaptivno ispitivanje uporabom računala imaju mnogo prednosti u usporedbi sa neadaptivnim ispitivanjem. Usprkos tome, broj sustava u kojima su implementirane metode za adaptivno ispitivanje nije velik što se zasigurno može pripisati nedostacima adaptivne metode. Implementacija te metode zahtjeva izuzetno mnogo vremena te se razvijanje adaptivnog ispitivanja najčešće isplati samo u sustavima koji ispituju jako velik broj ispitanika. U ovom radu predložen je i implementiran algoritam za adaptivno ispitivanje koji se temelji na iskorištavanju semantičkih veza iz hijerarhijske strukture segmenata znanja na kolegiju. Time je omogućeno korištenje adaptivnog ispitivanja i na manjim kolegijima koje pohađa nekoliko stotina studenata. Prilikom implementacije se posebno vodilo računa o jednostavnosti provedbe adaptivnog ispitivanja od strane nastavnog osoblja i količini vremena koje bi osoblje moralo utrošiti za uspješno provođenje ispitivanja.

Sustavi *StudTest2* i *Ferko* su nadograđeni kako bi podupirali provedbu adaptivnog ispitivanja. Kako se sustav *Ferko* koristi za administriranje velikog broja kolegija na Fakultetu elektrotehnike i računarstva, iskreno se nadam da će barem mali broj nastavničkog osoblja uvidjeti prednosti ovog načina ispitivanja te uložiti dodatan trud za moguće poboljšanje svojih kolegija korištenjem ove metode za provedbu ispitivanja na svojim predmetima.

7 Literatura

1 Glas, C.A.W., Linden, W.J. van der, *Computerized Adaptive Testing: Theory and Practice*, 1. izdanje, Dordrecht, Nizozemska: Springer, 2000. godina

2 Graduate Record Examination, http://www.ets.org/gre

3 Wikipedia, *Computerized classification test*, http://en.wikipedia.org/wiki/Computerized_classification_test

4 Russell, S., Norvig P., *Artificial Intelligence: A Modern Approach*, 2. izdanje, New Yersey: Pearson Education Inc., 2002. godina

5 *LOM Learning Object Metadata IEEE 1484.12 standard draft*, http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

6 US Department of Defense, Advanced Distributed Learning, http://www.adlnet.gov/

7 Čupić, M., Model sustava e-ispitivanja s potporom za inteligentno upravljanje provjerom znanja, magistarski rad, Fakultet elektrotehnike i računarstva, 2006. godina

8 Java Course Management System, http://morgoth.zemris.fer.hr/trac/jcms/wiki

9 Hibernate, http://www.hibernate.org/

10 Java Persistence API, http://wiki.oracle.com/page/JPA

11 Java Applets, http://java.sun.com/applets/

8 Sažetak

Web-temeljeni sustav za provjeru znanja studenta

U sklopu ovog diplomskog rada izvršena je usporedba neadaptivnog i adaptivnog ispitivanja uporabom računala. Predstavljen je teoretski okvir za provedbu adaptivnog ispitivanja uporabom računala te je osmišljen i implementiran algoritam za adaptivno ispitivanje. Definiran je način strukturiranja hijerarhije znanja na kolegiju koji se koristi u algoritmu. Sustavi StudTest2 i Ferko nadograđeni su kako bi podupirali provedbu e-ispitivanja na temelju osmišljenog algoritma za adaptivno ispitivanje.

Ključne riječi: *e-ispitivanje, adaptivno ispitivanje, semantičke mreže*

Web-based system for student knowledge assessment

In this thesis adaptive and non-adaptive assessment are compared. A theoretical overview of adaptive assessment method has been given and an algorithm based on it has been developed and implemented. A way to represent structured knowledge of a course is explained and that representation is being used by the algorithm. StudTest2 and Ferko systems have been upgraded so they can enforce e-assessment based on the developed algorithm for adaptive assessment.

Keywords: *e-assessment, adaptive assessment, semantic networks*