SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Marko Čupić

MODEL SUSTAVA E-ISPITIVANJA S POTPOROM ZA INTELIGENTNO UPRAVLJANJE PROVJEROM ZNANJA

MAGISTARSKI RAD

Zagreb, 2006.

Magistarski rad je izrađen u Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave, Fakulteta elektrotehnike i računarstva.

Mentor: prof. dr. sc. Vlado Glavinić

Magistarski rad ima: 149 stranica

Rad br.:

Povjerenstvo za ocjenu u sastavu:

1. Prof. dr. sc. Slavomir Stankov – Fakultet prirodoslovno matematičkih znanosti i odgojnih područja Split – predsjednik
2. Prof. dr. sc. Vlado Glavinić – Fakultet elektrotehnike i računarstva – mentor
3. Prof. dr. sc. Bojana Dalbelo-Bašić – Fakultet elektrotehnike i računarstva

Povjerenstvo za obranu u sastavu:

1. Prof. dr. sc. Slavomir Stankov – Fakultet prirodoslovno matematičkih znanosti i odgojnih područja Split – predsjednik
2. Prof. dr. sc. Vlado Glavinić – Fakultet elektrotehnike i računarstva – mentor
3. Prof. dr. sc. Bojana Dalbelo-Bašić – Fakultet elektrotehnike i računarstva

Datum obrane: 05. srpnja 2006.

Sadržaj

1 UVOD 1

2 SUSTAVI E-UČENJA I ODNOSNE NORME 6

2.1 Općenito o sustavima e-učenja 6
2.2 Modeli sustava e-učenja 9
2.3 Norme u e-učenju 11
2.4 Norma LOM 13
2.5 Norma SCORM 15
2.6 Norma "IMS Question and Test Interoperability" 18

3 E-ISPITIVANJE U POSTOJEĆIM SUSTAVIMA 24

3.1 Osvrt na mogućnosti sustava WebCT 24
3.2 Osvrt na mogućnosti sustava MOODLE 25
3.3 Kratka analiza 27
3.4 Sustav WODLS 27
3.5 Sustav CMS Nescume 29

4 MODEL E-ISPITIVANJA STUDTEST 32

4.1 Zahtjevi na sustav e-ispitivanja 32
4.2 Ideja višestruke iskoristivosti zadataka 34
4.3 Pisanje provjere znanja 35
4.4 Koncepti modela StudTest 38
4.5 Komponente za potporu dinamičkoj podršci izrade višemedijskih sadržaja 53
4.6 Potpora tehnologijama Semantičkog Weba 53

5 MODEL INTERAKCIJE KOMPONENTI SUSTAVA E-ISPITIVANJA STUDTEST 55

5.1 Stvaranje novog opisnika provjere znanja 55
5.2 Konfiguriranje podesivih komponenti 56
5.3 Poslovanje provjerom znanja 57
5.3.1 Početno stvaranje provjere znanja 57
5.3.2 Pisanje provjere znanja 57
5.3.3 Ocjenjivanje provjere znanja 59
5.4 Zahtijevanje pokretanja testa 60
5.5 Stvaranje novog primjerka zadatka na temelju zadatka 60

6 POTPORA INTELIGENTNOM VOĐENJU E-ISPITIVANJA 61

6.1 Ontološki opis gradiva kolegija 61
6.2 Proširenje ontologije feronto1 65
6.3 Komponente za inteligentno vođenje ispitnog procesa 66

7 IMPLEMENTACIJA 69

7.1 Implementacije koncepata modela StudTest 71
7.1.1 Upravljači provjere znanja 71
7.1.2 Provjerivači dozvole početka pisanja provjere znanja 73
7.1.3 Nadglednici tijeka pisanja provjere znanja 74
7.1.4 Ocjenjivači 75
7.1.5 Uporaba na kolegijima 76
7.1.6 Implementirani tipovi zadataka 79
7.2 Uporaba ontologija u postojećem sustavu 86
7.2.1 Potpora radu s ontologijama 86
7.2.2 Ontologija StudTest-Zadaci 88
7.2.3 Uporaba ontologije StudTest-Zadaci 88

8 PRIMJERI RAZVIJENIH ZADATAKA 91

8.1 Statičke ABC-pitalice 91

8.1.1 Statička pitanja s jednim točnim odgovorom ... 91
8.1.2 Statička pitanja s više točnih odgovora ... 92
8.2 DINAMIČKE ABC-PITALICE ... 93
8.2.1 Dualna funkcija ... 93
8.2.2 Komplementarna funkcija ... 93
8.2.3 Rezultat izračuna izraza jezika VHDL ... 93
8.3 UNOS TEKSTA ... 94
8.3.1 Algebarski zapis minterma/maksterma ... 94
8.3.2 Minimizacija funkcije ... 94
8.3.3 Rezultat simulacije modela VHDL ... 94
8.4 UNOS LISTE ... 95
8.4.1 Minimizacija Quine-McClusky ... 95
8.4.2 Programiranje sekvencijskog sklopa preglednim tablicama ... 96
8.5 PITANJA SLOŽENOG GRAFIČKOG KORISNIČKOG SUČELJA ... 97
8.5.1 Minimizacija Quine-McClusky ... 97
8.5.2 Programiranje sklopa PLA ... 98
8.5.3 Implementacija funkcije tehnologijom CMOS ... 98
8.6 UPORABA POMOĆNIKA ... 99
8.6.1 Očitavanje sadržaja memorije izvedene diodama i tehnologijom MOSFET ... 99
8.6.2 Očitavanje ciklusa sekvencijskog sklopa ... 101

9 ZAKLJUČAK ... 103

10 PRILOZI ... 108

10.1 KONSTANTE I POMOĆNI OBJEKTI MODELA STUDTEST ... 109
10.2 PRIMJERI RAZVIJENIH PRLETA ... 123

Popis slika

SLIKA 2-1. POJEDNOSTAVLJEN POGLED NA EVOLUCIJU SUSTAVA E-UČENJA 7
SLIKA 2-2. KLASIFIKACIJA SUSTAVA M-UČENJA 8
SLIKA 2-3. FUNKCIJSKI MODEL SUSTAVA E-UČENJA PREMA [9] 9
SLIKA 2-4. FUNKCIJSKI MODEL SUSTAVA E-UČENJA PREMA EDUWORKS 10
SLIKA 2-5. MODEL SUSTAVA E-UČENJA TEMELJEN NA WEB USLUGAMA [9] 11
SLIKA 2-6. MODEL LMS SUSTAVA PREMA NORMI SCORM 16
SLIKA 2-7. KONCEPTUALNI POGLED NA PAKET JEDINICE MATERIJALA ZA UČENJE 17
SLIKA 2-8. QTI POGLED NA PROCES PROVJERE ZNANJA. UML NOTACIJA. 19
SLIKA 3-2. ODNOS SUSTAVA NESCUME I STUDTEST 30
SLIKA 4-2. PROVJERA ZNANJA U SUSTAVU E-ISPITIVANJA 37
SLIKA 4-3. MEĐUSOBNI ODNOS DIJELA KONCEPATA MODELA STUDTEST 39
SLIKA 4-4. PRIMJER ZADATKA S JEDNIM TOČNIM ODGOVOROM 41
SLIKA 4-5. PRIMJER ZADATKA S VIŠE TOČNIH ODGOVORA 41
SLIKA 4-6. PRIMJER ZADATKA S UNOSOM TEKSTA 42
SLIKA 4-7. HTML KOD ZADATKA S UNOSOM TEKSTA 43
SLIKA 4-8. ODNOSI IZMEĐU DIJELA OSNOVNIH OBJEKATA MODELA 47
SLIKA 4-9. GRAĐA PRLETA 48
SLIKA 5-1. PRIMJER SUČELJA ZA PODEŠAVANJE PARAMETARA OPISNIKA PROVJERE ZNANJA 56
SLIKA 5-2. UPRAVLJANJE TIJEKOM PROVJERE ZNANJA 58
SLIKA 6-1. ONTOLOŠKI PRIKAZ DIJELA KONCEPATA DIGITALNE ELEKTRONIKE 63
SLIKA 6-2. DOPUNA ONTOLOŠKOG OPISA KOLEGIJA VEZAMA SA ZADACIMA 66
SLIKA 6-3. SVE POZNATO O C:[REGISTERS] 67
SLIKA 7-1. GRAĐA PROTOTIPNE IMPLEMENTACIJE SUSTAVA STUDTEST 70
SLIKA 7-2. UPORABA SUSTAVA STUDTEST KROZ SUSTAV E-UČENJA 70
SLIKA 7-3. SUČELJE ZA PODEŠAVANJE PARAMETARA PISANJA PROVJERE ZNANJA 72
SLIKA 7-4. PRIMJER OCJENJIVAČA S NEGATIVNIM BODOVIMA ZA NETOČNE I NERIJEŠENE ZADATKE 75
SLIKA 7-5. PRIMJER OCJENJIVAČA KOJI NEGATIVNE BODOVE DAJE SAMO ZA NETOČNE ODGOVORE 75
SLIKA 7-6. OCJENJIVAČ KOJI DAJE BROJ BODOVA PROPORCIONALAN MJERI TOČNOSTI 76
SLIKA 7-7. OCJENJIVAČ S BROJEM BODOVA PROPORCIONALNIM MJERAMA TOČNOSTI I POUZDANOSTI 76
SLIKA 7-8. OCJENJIVAČ KOJI DAJE RAZLIČIT BROJ BODOVA OVISNO O OCJENJIVANOM ZADATKU 76
SLIKA 7-12. SUČELJE ZA POTPORU RADU S ONTOLOGIJAMA 87
SLIKA 7-13. PRIMJER HIJERARHIJSKOG RAZMJEŠTANJA ZADATAKA 88
SLIKA 7-15. PRIMJER UPORABE ONTOLOGIJE ZA ODABIR ZADATAKA 90
SLIKA 8-1. PRIMJER ZADATKA 91
SLIKA 8-2. PRIMJER ZADATKA 92
SLIKA 8-3. PRIMJER ZADATKA 92
SLIKA 8-4. PRIMJER ZADATKA 93
SLIKA 8-5. PRIMJER ZADATKA 93
SLIKA 8-6. PRIMJER ZADATKA 94
SLIKA 8-7. PRIMJER ZADATKA 94
SLIKA 8-8. PRIMJER ZADATKA 94
SLIKA 8-9. PRIMJER ZADATKA 95
SLIKA 8-10. PRIMJER ZADATKA 96
SLIKA 8-11. PRIMJER ZADATKA 97
SLIKA 8-12. PRIMJER ZADATKA 98
SLIKA 8-13. PRIMJER ZADATKA 98
SLIKA 8-14. PRIMJER ZADATKA 99
SLIKA 8-15. PRIMJER ZADATKA 100
SLIKA 8-16. PRIMJER ZADATKA 101
SLIKA 8-17. PRIMJER ZADATKA 102

Popis tablica

TABLICA 2-1. KATEGORIJE PODATAKA OSNOVNE SHEME LOM VERZIJE 1.0 13
TABLICA 2-2. PODACI DEFINIRANI U OKVIRU KATEGORIJE GENERAL NORME LOM 14
TABLICA 3-1. USPOREDBA SUSTAVA WEBCT, MOODLE, WODLS I STUDTEST 31
TABLICA 4-2. ELEMENTI KONCEPTA TEST 40
TABLICA 4-4. ELEMENTI KONCEPTA PROBLEMTYPE 42
TABLICA 4-5. METODE KOJE ZADATAK MORA IMPLEMENTIRATI KAO PODRŠKU TIPU ZADATKA 42
TABLICA 4-6. METODE PRIKAZIVAČA ZADATKA (PROBLEMRENDERER) 43
TABLICA 4-7. METODE PRIKAZIVAČA PROVJERE ZNANJA 44
TABLICA 4-8. ELEMENTI KONCEPTA PROBLEMGENERATOR 44
TABLICA 4-9. METODE UREĐIVAČA ZADATKA 45
TABLICA 4-10. ELEMENTI KONCEPTA PROBLEM 45
TABLICA 4-11. ELEMENTI KONCEPTA PROBLEMINSTANCE 46
TABLICA 4-12. METODE STVARATELJA PRIMJERKA ZADATKA 47
TABLICA 4-13. METODE VREDNOVATELJA PRIMJERKA ZADATKA 47
TABLICA 4-14. ELEMENTI KONCEPTA TESTSTARTQUEUE 48
TABLICA 4-15. ELEMENTI KONCEPTA TESTSTARTREQUEST 49
TABLICA 4-16. METODE OCJENJIVAČA PRIMJERKA ZADATKA 49
TABLICA 4-17. METODE PODESIVIH KOMPONENTI 49
TABLICA 4-18. METODE PROVJERIVAČA DOZVOLE POČETKA PISANJA PROVJERE ZNANJA 50
TABLICA 4-19. METODE NADGLEDNIKA TIJEKA PISANJA PROVJERE ZNANJA 51
TABLICA 4-21. METODE POMOĆNIKA 53
TABLICA 5-1. PARAMETRI KOMPONENTE TESTDESCRIPTOR 55
TABLICA 6-1. NEKI KONCEPTI ONTOLOGIJE FERONTO1 61
TABLICA 6-2. NEKE RELACIJE ONTOLOGIJE FERONTO1 62
TABLICA 6-3. DEFINICIJE KORIŠTENIH POKRATA KOD URI-JA 63
TABLICA 6-4. PROŠIRENJE ONTOLOGIJE FERONTO1 NOVIM KONCEPTIMA 65
TABLICA 6-5. PROŠIRENJE ONTOLOGIJE FERONTO1 NOVIM RELACIJAMA 65
TABLICA 7-1. PARAMETRI KOJE NUDI KOMPONENTA SIMPLETESTCONTROLLER 72
TABLICA 7-2. SUČELJE IABCSINGLECORRECTTYPE 80
TABLICA 7-3. SUČELJE IABCSINGLECORRECTTYPEEDITOR 80
TABLICA 7-4. SUČELJE IABCMULTICORRECTTYPE 81
TABLICA 7-5. SUČELJE IABCMULTICORRECTTYPEEDITOR 81
TABLICA 7-6. SUČELJE ITEXTTYPE 82
TABLICA 7-7. SUČELJE ITEXTTYPEEDITOR 82
TABLICA 7-8. SUČELJE ITEXTLISTTYPE 83
TABLICA 7-9. SUČELJE ITEXTLISTTYPEEDITOR 84
TABLICA 7-10. SUČELJE ICUSTOMPROBLEMPANELTYPE 85
TABLICA 7-11. SUČELJE ICUSTOMPROBLEMPANELTYPEEDITOR 86
TABLICA 10-1. VRIJEDNOSTI ELEMENTA TESTINSTANCE.STATUS 110
TABLICA 10-2. VRIJEDNOSTI ELEMENTA TESTDESCRIPTOR.AUTOCREATIONPOLICY 111
TABLICA 10-3. VRIJEDNOSTI ELEMENTA PROBLEMINSTANCE.GRADINGSTATUS 112
TABLICA 10-4. VRIJEDNOSTI ELEMENTA PROBLEMINSTANCE.EVALSTATUS 113
TABLICA 10-5. VRIJEDNOSTI ELEMENTA PROBLEMINSTANCE.PROBLEMINSTANTIATIONSTATUS 114
TABLICA 10-6. METODE POMOĆNOG OBJEKTA ITESTSTARTCHECKERSUPPORT 115
TABLICA 10-7. METODE POMOĆNOG OBJEKTA ITESTSUPERVISORSUPPORT 116
TABLICA 10-8. STATUSI KOJE VRAĆA METODA TESTCONTROLLER.ONACTION 117
TABLICA 10-9. METODE KOJE MORA IMPLEMENTIRATI SVAKA KOMPONENTA IREPOSITORIES 118
TABLICA 10-10. METODE KOJE MORA IMPLEMENTIRATI SVAKA KOMPONENTA IREPOSITORY 119
TABLICA 10-11. METODE KOJE MORA IMPLEMENTIRATI SVAKA KOMPONENTA IKEYREPOSITORY 120
TABLICA 10-12. METODE KOJE MORA IMPLEMENTIRATI IATTACHMENTREPOSITORY 121
TABLICA 10-13. METODE KOJE MORA IMPLEMENTIRATI SVAKA KOMPONENTA IATTACHMENT 122

1 Uvod

U današnje doba sve je učestalija uporaba računala u obrazovnom procesu. Razvojem računarstva i procesne moći računala, uporaba računala polako se pomiče iz područja nadopune nastavnom procesu ka području gdje računala uče ljude (primjerice inteligentni tutorski sustavi), odnosno omogućavaju pristup uporabom Interneta kroz različite pristupne tehnologije (počev od stolnih računala pa sve do podrške korisnicima mobilnih uređaja – što čini sustave m-učenja).

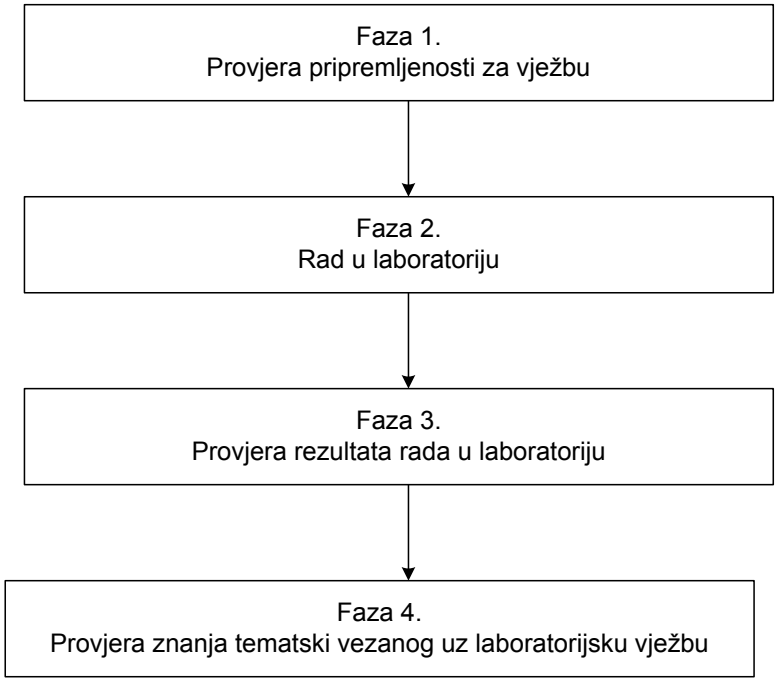
Pogledamo li kako se sve računala koriste kao nadopuna klasičnoj nastavi, dolazimo do sustava za prikupljanje i analizu radova studenata (pomoć samoj administraciji kolegija te laboratorijskim vježbama npr. Nescume [24]), sustava za obavljanje laboratorijskih vježbi kroz virtualne instrumente (npr. LabView [40]) te sustava za simulaciju modela razrađenih na nastavi, kako bi se stekao dublji uvid u predstavljene koncepte [41] [42]. Od sustava čija je namjena vođenje studenata kroz obrazovni proces i pružanje nastavnog materijala, možemo spomenuti klasične predstavnike poput WebCT [44] i MOODLE [45], a može se uključiti i sustav poput WODLS [25], koji predstavlja skromniju varijantu prethodna dva sustava. Problem ovih sustava je loše riješen podsustav provjere znanja studenata, koji se svodi na niz elementarno jednostavnih podržanih vrsta pitanja. Ovaj rad nastao je kao pokušaj rješavanja upravo ovog problema, a to je ponuditi sustav e-ispitivanja, koji će ujedno sadržavati i potporu za inteligentno vođenje ispitnog procesa, odnosno ponuditi jednostavnu izradu adaptivnih provjera znanja. Konačno, od sustava koji su u mogućnosti samostalno voditi studente kroz postupak usvajanja znanja, i koji su nastali kroz psihološke studije usmjerene na pitanja poput "Kako čovjek uči?", potrebno je spomenuti inteligentne tutorske sustave, primjerice TexSys [8].

Na Fakultetu elektrotehnike i računarstva održava se niz masovnih kolegija, koje pohađa između 300 i 1200 studenata. Organizacija cjelokupnog nastavnog procesa za ovakve je kolegije izuzetno složeno pitanje, jer je potrebno osigurati i ljudske resurse (predavače, asistente), prostorne resurse (dvorane za održavanje predavanje, laboratorije za izvođenje laboratorijskih vježbi, te dvorane za izvođenje provjera znanja), te materijalne resurse (oprema za laboratorije, računala i sl.).

Analiziramo li primjerice organizaciju laboratorijskih vježbi, dolazimo do zaključka da se za sve laboratorijske vježbe studenti kod kuće moraju samostalno pripremiti (kako bi znali što trebaju napraviti u laboratoriju, te kako bi se spriječile po život opasne situacije u kojima se radi primjerice s visokim naponom) te potom na vježbama obaviti postavljene zadatke. Problem provjere pripremljenosti studenata, kao i bodovanja njihovog rada na laboratoriju tipično se rješava pregledom priprema od strane asistenata, te pred kraj vježbe provjerom uspješnosti rješavanja postavljenih zadataka, odnosno provjera razumijevanja odgovarajuće teorijske podloge vezane uz laboratorijske vježbe. Slika 1-1 daje malo detaljniji prikaz ove procedure.

U fazi 1 provjerava se pripremljenost studenta za laboratorijsku vježbu – npr. pregledavanjem priprema te ispitivanjem o pripremi i njenom sadržaju, kako bi se otkrile prepisane pripreme.

U fazi 2 studenti samostalno obavljaju laboratorijsku vježbu, uz konzultiranje asistenata za detalje koje nisu u stanju samostalno riješiti. Iako bi ovo zapravo trebao



Slika 1-1. Izvođenje laboratorijske vježbe

biti glavni razlog "boravka" asistenata na laboratorijskim vježbama, tablica 1-1 za ovo ne predviđa niti minute vremena.

U fazi 3 studenti predaju rezultate rada na provjeru. Ukoliko se vježba obavlja korištenjem računala, asistent provjerava je li student uspješno obavio zadatak. Nakon provjere rješenja, student je obično dužan elektronički oblik rješenja dostaviti asistentu, radi arhiviranja, naknadnih provjera, otkrivanja plagijata i sl.

Faza 4 predviđa provjeru znanja studenta kojom se želi otkriti je li student samostalno izradio rješenje (ili ga je prepisao od kolege) te koliko student ima teorijskog znanja vezanog uz vježbu koju je upravo obavio. Ova faza u pojedinim

Tablica 1-1. Podaci za laboratorijske vježbe nekih kolegija

	Digitalna elektronika	Mreže računala	Digitalna logika
Broj studenata	600	300	1100
Broj grupa po tjednu	20	10	54
Studenata u grupi	30	30	20
Broj termina u semestru	8	15	3
Trajanje termina	180 min	90 min	240 min
Broj asistenata na kolegiju	10	5	39
Broj asistenata po terminu	2	2	~1.33
Procijenjeno vrijeme potrebno za provjeru jednog studenta	15 min	15 min	15 min
Vrijeme dopušteno trajanjem termina za provjeru jednog studenta	12 min	6 min	~16 min
Razlika (neiskorišteno vrijeme)	-3 min	-9 min	+1 min

kolegijima može studentima nositi bodove koji direktno utječu na završnu ocjenu kolegija.

Ovakav pristup ima svojih prednosti, ali i velikih mana. Prednost pristupa jest mogućnost da asistent uživo razgovara sa svakim studentom, i temeljno provjeri njegovo znanje. Ukoliko to vrijeme dopušta, ovakav pristup također omogućava i asistentu da studentu objasni pojmove koji nisu dobro (ili uopće) shvaćeni.

Negativne strane ovakvog pristupa sve redom proizlaze iz masovnosti samih kolegija. Prvi veliki problem jest opterećenje asistenata, pri čemu zbog izuzetno velikog broja studenata jedan asistent ne može sam obaviti cjelokupni posao vođenja laboratorijskih vježbi i provjere znanja studenata. Umjesto toga, za laboratorij je tipično zaduženo više asistenata koji nužno nemaju iste kriterije. Zbog toga se događa da neki asistenti studente pitaju vrlo površno a neki vrlo detaljno; neki dobiju puno bodova za malo iskazanog znanja, a neki malo bodova uz puno iskazanog znanja – što povlači da isto znanje studenta kod različitih asistenata biva bitno drugačije ocijenjeno.

Drugi veliki problem ovog pristupa opet proizlazi iz velikog broja studenata i velikog broja kolegija na kojima radi pojedini asistent što za posljedicu ima premali broj ljudi unutar Zavoda koji drži kolegij, a taj se manjak u posljednje vrijeme nadomješta iz fakultetskih resursa neopterećenih asistenata (engl. pools). Ovo za posljedicu ima stvaranje konačnog skupa asistenata koji vode laboratorijske vježbe ovako masovnih kolegija od kojih su neki asistenti vrlo zainteresirani, a neki jedva da se sjećaju što se na dotičnom kolegiju radi, i na takav način vode i same laboratorijske vježbe.

Ukoliko se krene u analizu kako bi trebao izgledati ogledni termin laboratorijskih vježbi, dolazimo do sljedećeg scenarija. Po ulasku studenata u laboratorij, a prije no što krenu same vježbe, asistent svim studentima detaljno pregledava pripreme. Studenti koji imaju neodgovarajuće pripreme ili ih čak nisu niti sami napisali udaljavaju se sa vježbe. Potom studenti obavljaju laboratorijsku vježbu. Pred kraj termina određenog za laboratorijske vježbe, asistenti ponovno obilaze cijeli laboratorij i provjeravaju za svakog studenta što je točno napravio, je li to sam napravio te da li je vježba uspješno obavljena. Na temelju ove provjere, asistent svakom studentu dodjeljuje određen broj bodova, u skladu s politikom samog kolegija i dogovorenim načelima bodovanja. Tablica 1-1 ilustrira kako ovo točno izgleda na konkretnim primjerima kolegija Digitalna elektronika, Mreže računala te Digitalna logika, koji se predaju na Fakultetu elektrotehnike i računarstva. Podatak o procijenjenom vremenu potrebnom za provjeru znanja jednog studenta dobiven je kroz višegodišnje iskustvo u držanju laboratorijskih vježbi na tim kolegijima.

Pogledamo li najkritičniji primjer (Mreže računala), za kvalitetno obavljanje provjere izvedbe laboratorijske vježbe potrebno je oko 15 minuta po studentu. Naime, na tom kolegiju dosta studenata ne rješava vježbu samostalno, jer je za uspješan završetak vježbe potrebno napisati dosta složene programe u programskom jeziku C, koji obavljaju komunikaciju na podatkovnom, mrežnom, prijenosnom ili aplikacijskom sloju referentnog modela ISO/OSI. Problem nastaje kada student ustvrdi da je samostalno napisao taj kod, a nije. Ovo je moguće i demonstrirati, čak i samom studentu, tako da se student dovede do situacije u kojoj niti on sam ne zna odgovoriti na pitanje što bi njegov program u određenom slučaju napravio, ili ako to zna, onda ne zna u izvornom kodu pronaći naredbe odgovorne za takvo ponašanje. U slučaju

ovog kolegija, nažalost, raspoloživo vrijeme koje se može utrošiti na provjeru znanja studenta iznosi samo 6 minuta po studentu, čime se asistenti prisiljavaju na nekvalitetno obavljanje svog posla i površno ispitivanje studenata.

Moguće rješenje ovog problema predstavlja efikasan način provjere znanja studenata, i to istovremeno, nad cijelom grupom. Ovo je jedino moguće izvesti kod kolegija koji imaju na raspolaganju laboratorije opremljene računalima te uz odgovarajuću programsku podršku, koja je u stanju obaviti kvalitetnu provjeru znanja – što je zadatak kvalitetnog sustava e-ispitivanja. Osim toga, na ovaj se način može u potpunosti eliminirati i utjecaj različitosti asistenata na vrednovanje rada studenata u laboratoriju.

Drugi interesantan primjer predstavlja izazov koji je Bolonjski proces postavio pred asistente, a to su domaće zadaće. Naime, prema Bolonjskom procesu, koji je na Fakultetu elektrotehnike i računarstva po prvi puta uveden akademske godine 2005/2006, sastavni dio elemenata ocjenjivanja svakog kolegija postale su i domaće zadaće.

Interesantno je pitanje svrhe bodovanja domaćih zadaća. Naime, tipični je scenarij zadavanja domaće zadaće situacija gdje se na predavanjima definira zadatak ili njih nekoliko koje studenti trebaju riješiti do sljedećeg sata ili nekog drugog roka. Studenti odlaze kući, netko zadatak riješi, većina prepiše rješenja, i u konačnici nastavnik boduje koliko je uspješno student prepisao rješenja. Ovakav oblik zadaća zapravo direktno potiče studente na organiziranje u svrhu prepisivanja, odnosno na timski rad u svrhu stjecanja koristi na nepošteni način.

Oblik domaće zadaće koji bi umanjio ovakva udruživanja predstavlja zadaća u kojoj svaki student dobije različita pitanja. U tom slučaju prepisivanje od kolege očito nije moguće, jer kolega nema takav zadatak. Međutim, boljim studentima uvijek se može platiti da riješe zadatak koji neki student ne zna riješiti. Ovo je evidentno pravi pristup za uvođenje studenata u moderne ekonomske tekovine, i poticanje studentskog poduzetništva. Međutim, uz pretpostavku da ova pojava neće biti prevelikog obima, što obzirom na loše ekonomsko stanje većine studenata i nije neopravdana pretpostavka, možemo reći da pogodnost suzbijanja sirovog prepisivanja i poticanje na samostalni rad ipak predstavljaju veliki dobitak. Posao ispravljanja studentskih zadaća u ovom je slučaju izuzetno složen, jer niti nastavno osoblje nema gotova rješenja, već svaki zadatak treba ispravljati zasebno. Ovakav je pristup, nažalost, moguć na kolegijima s 5 do 10 studenata. Kolegiji koji imaju 30 do 100 puta više studenata ne mogu direktno koristiti ovakav pristup, jer ispravljanje zadaće koja ima po 10 zadataka za 1000 studenata onda može trajati tjednima.

Kako bi se iskoristile dobre strane domaćih zadaća, očito je potrebno svim studentima zadati zadatke koji se razlikuju, tako da direktno prepisivanje nije moguće, a istovremeno je potrebno osigurati ispravljanje i vrednovanje tih zadaća u realnom vremenu, odnosno u što je moguće kraćem roku, a svakako prije zadavanja sljedeće domaće zadaće. Moguće rješenje ovog problema također predstavlja kvalitetno izvedeno sustav e-ispitivanja, koji bi riješio posao izrade različitih zadataka za različite studente kao i njihovo automatsko vrednovanje.

Temeljeći se na zahtjevima ovako postavljenog okvira, u radu se opisuje modeliranje, prototipna izvedba i vrednovanje "u živo" sustava za e-ispitivanje s potporom za inteligentno vođenje provjere znanja StudTest.

Ovaj rad organiziran je na sljedeći način. U poglavlju 2 dan je kratak uvod u sustave e-učenja, i trenutno važeće norme na ovom području. U poglavlju se definira sustav e-učenja te se razmatraju njegove podvrste. Potom se kratko uvode norme IEEE LOM i SCORM koje se direktno odnose na sustave e-učenja. Na kraju, slijedi opis norme IMS Q&TI, koja predstavlja jedan pokušaj normiranja samog područja e-ispitivanja.

U poglavlju 3 ukratko se razmatraju mogućnosti danas popularnih sustava e-učenja (WebCT, MOODLE) te lokalno razvijenog sustava WODLS u okviru podrške e-ispitivanju. Dan je i kratak osvrt na sustav Nescume, koji se koristi za pristup sustavu e-ispitivanja razvijenog i opisanog u ovom radu.

Poglavlje 4 uvodi model sustava StudTest, i definira osnovne koncepte tog modela. Ukratko je opisana uloga definiranih koncepata i njihov međuodnos.

Poglavlje 5 definira način interakcije između pojedinih implementacija koncepata odgovarajućih modela, čime se ujedno definiraju i njihova zaduženja.

Poglavlje 6 donosi osvrt na mogućnost uporabe ovako definiranih koncepata za izvedbu sustava koji omogućava inteligentno (ili adaptivno) vođenje ispitnog procesa. Ukratko se ilustrira što bi trebalo napraviti i na koji način, kako bi se pružila ovakva usluga.

Poglavlje 7 opisuje prototipnu implementaciju sustava e-ispitivanja zasnovanu na opisanom modelu StudTest. Opisuje se korištena tehnologija i razlozi njene uporabe, te konkretne implementacije koncepata samog modela.

Poglavlje 8 ilustrira mogućnosti razvijenog sustava. Poglavlje sadrži niz razvijenih zadataka, uz kratke komentare o mogućnostima zadataka, načinu postavljanja pitanja i načinu vrednovanja studentovih odgovora.

Poglavlje 9 donosi zaključak rada. Iznosi se kritički osvrt na razvijeni model, njegove karakteristike, način primjene sustava u stvarnoj nastavi te postignuti rezultati.

Rad sadrži i dodatak, u kojem se nalazi najveći dio zadataka razvijenih za potrebe kolegija Digitalna logika, čime se do kraja ilustrira prava snaga ovakvog pristupa problematici e-ispitivanja.

2 Sustavi e-učenja i odnosne norme

Paralelno s klasičnim oblikom održavanja nastave oduvijek je postojala potreba i za učenjem na daljinu, odnosno oblicima učenja gdje učitelj nije direktno na raspolaganju za živu komunikaciju. Prva primjena učenja na daljinu na europska je sveučilišta uvedena početkom 20. stoljeća kroz dopisne programe [1]. Kasnih 50-ih za potrebe učenja na daljinu započela je uporaba radija i televizije koji su omogućavali istovremeni prijenos informacija i nastavnog materijala većem broju polaznika. Ovaj pristup pokazao je nekoliko problema, od kojih je najveći nemogućnost dvosmjerne komunikacije na razini učitelj-učenik. Taj nedostatak automatski je povlačio lošiju kvalitetu obrazovanja te uporabu ovih tehnologija samo kao dopunu klasičnoj nastavi, što je tijekom 70-ih dovelo do povratka na klasični način poučavanja.

Tijekom 80-ih dolazi do novog obrata uzrokovanog razvojem tehnologije. Populariziraju se "oglasne ploče" (BBS, engl. Bill-Board Systems) te način komunikacije uporabom elektroničke pošte (engl. e-mail). Iako je sam razvoj ovih tehnologija išao relativno sporo, u kombinaciji s pojavom tehnologija Weba dolazi do masovnog prihvaćanja ovih tehnologija i njihove uporabe za sustave učenja na daljinu (što je naročito postalo izraženo prihvaćanjem i uporabom na sveučilištima diljem Europe, Amerike i Australije. U 90-ima tako nastaje i poznat termin "e-učenje" (engl. e-learning), jer Internet postaje medij koji omogućava isporuku sadržaja za učenje i dvosmjernu komunikaciju. Ipak, još i danas se provlače neka važna pitanja: može li e-učenje i poučavanje biti efektivno kao što je klasično učenje i poučavanje, te je li nam to uopće potrebno?

U okviru ovog rada e-učenje će biti definirano na sljedeći način [10].

> *E-učenje podrazumijeva isporuku obrazovnih programa i/ili programa za vježbanje i/ili materijala elektroničkim putem. E-učenje uključuje uporabu računala ili drugih elektroničkih uređaja (primjerice mobilnih telefona) pri čemu se raspodjela materijala može vršiti sinkrono ili asinkrono uporabom tehnologija za umrežavanje računala.*

2.1 Općenito o sustavima e-učenja

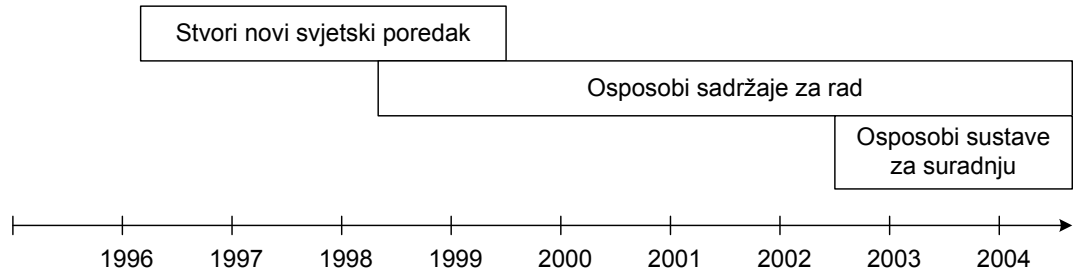
Intenzivni razvoj sustava e-učenja započinje sredinom devedesetih. Uloga ovih sustava jest pružiti obrazovanje svima, svugdje i u svako doba, te po prihvatljivoj cijeni. Kroz razvoj sustava e-učenja očekuje se povećanje sveopće obrazovanosti stanovništva, jer se kroz sustave e-učenja efikasnije može raditi s velikim brojem polaznika. Pojednostavljen pogled na evoluciju normi e-učenja prikazuje Slika 2-1 [4].

Na početku razvoja sustava e-učenje može se uočiti jedan entuzijazam ("Stvori novi svjetski poredak" – poredak u kojem računala samostalno obrazuju široke populacije ljudi, i sve radi idealno) koji je rezultirao naglim zanimanjem za sustave e-učenja i njihovu popularizaciju – međutim, područje još nije bilo dobro razrađeno i istraženo, što je imalo za posljedicu velika očekivanja – i izostanak njihovog ispunjenja. Jedan od glavnih razloga ovome jest očekivanje da će sustavi e-učenja moći u potpunosti i jednako kvalitetno odigrati ulogu živog učitelja, i sve što je vezano uz živog učitelja (odabir sadržaja za učenje, vođenje procesa učenja, diskusija s učenikom, provjera što i koliko kvalitetno je učenik naučio i slično). Međutim, da bi se to moglo

ostvariti, takvi bi sustavi nužno trebali imati komponentu umjetne inteligencije, koja bi sustavu omogućila da shvati i pojmi stupanj učenikovog savladavanja gradiva, način na koji on najbolje uči (primjerice, neki bolje uče kroz konkretne primjere, a neki bolje uče kroz apstraktne pojmove), probleme koje učenik ima s gradivom i sl. Primjerice, studije pokazuju da [5, p.p. SCORM 1-9]:

- brzina kojom pojedinci mogu usvajati određenu količinu gradiva varira za faktor od 3 do 7 – čak i u slučaju pažljivo odabranih studenata [5],
- u prosjeku, učenik u razredu pita 0,1 pitanje po satu [6],
- u direktnom radu s tutorom (učenje "jedan-na-jedan") učenik može pitati ili odgovoriti čak i na 120 pitanja po satu [6],
- uspjeh učenika koji su imali osobnog tutora može premašiti uspjeh učenika uključenog u razrednu nastavu čak za dvije standardne devijacije [7].

Ovi rezultati jasno upućuju na prednosti koje ima direktan rad s učenikom u obliku tutoriranja, što je uloga predviđena za moderne sustave e-učenja. Nažalost, danas još uvijek na polju umjetne inteligencije nema zadovoljavajućih rješenja koji bi ovo omogućili (iako se polako ali sigurno približavamo tom cilju). Danas i u klasičnom obrazovnom procesu prevladava razredna nastava, pri čemu grupe učenika ili studenata mogu varirati od 20 pa do 200 i više. Tutorska nastava nije toliko zastupljena jer zahtjeva prevelike resurse u nastavnom kadru, prostoru, vremenu i cijeni.



Slika 2-1. Pojednostavljen pogled na evoluciju sustava e-učenja

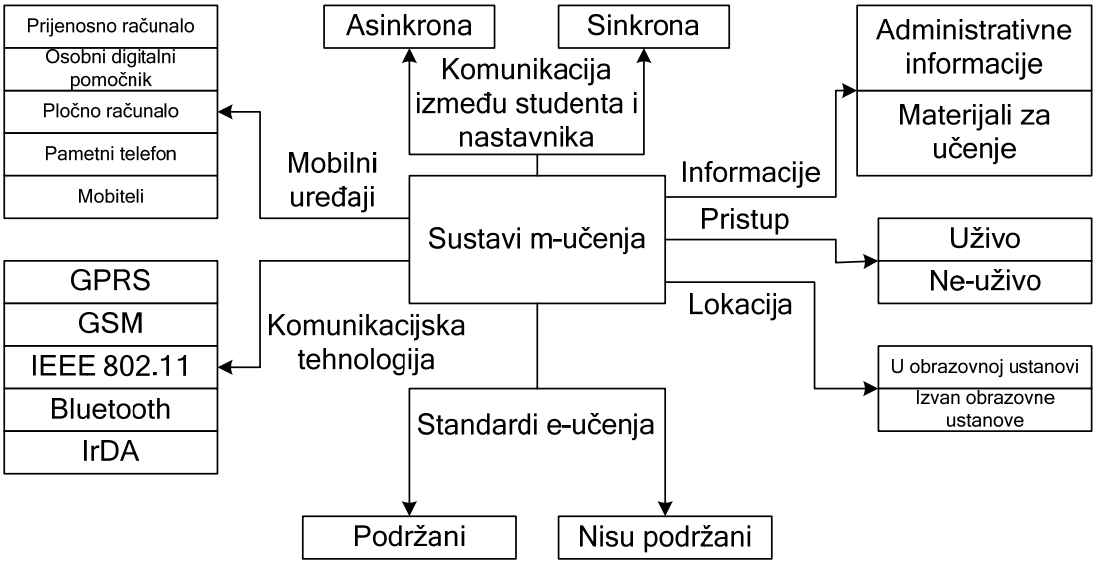
Zahvaljujući prednostima tutorskog oblika poučavanja (koje su već odavno poznate), još 60-ih godina prošlog stoljeća jedan smjer razvoja sustava za poučavanje krenuo je od tadašnjih istraživanja u polju umjetne inteligencije (i zaključivanja temeljenog na pravilima), te proučavanja načina na koji ljudi uče (i odgovarajućih kognitivnih modela). Ovaj smjer danas rezultira sustavima poznatim pod nazivom inteligentni tutorski sustavi (ITS, engl. intelligent tutoring system). Pri tome se pojam inteligentni odnosi na funkcionalnosti koje ovakav sustav mora imati:

- generiranje instrukcija u stvarnom vremenu i na zahtjev, prema potrebama individualnog učenika,
- podrška razgovoru koji može započeti učenik ili stroj, koji može biti slobodnog stila.

Osim što mogu biti utemeljeni na pravilima, današnji ITSovi sve se više temelje i na ontološkim opisima područnog znanja (primjerice, [8]).

Prema načinu podržanog učenja sustavi e-učenja mogu se podijeliti na sustave koji podržavaju [2]:

- *sinkrono učenje* podrazumijeva predavanje instruktora o specifičnoj temi uživo (engl. online), najčešće uporabom videokonferencija i tehnologija za sinkronu komunikaciju (razgovor – engl. chat, bijelu ploču – engl. whiteboard i sl.); to ujedno znači i da svi sudionici ovog oblika učenja moraju u određeno vrijeme biti slobodni i naći se na zadanom mjestu u zadano vrijeme,
- *asinkrono učenje* podrazumijeva samostalno učenje uporabom edukacijskih materijala i zabilješki s prethodnih predavanja ili zapisnika s prethodnih suradnih sjednica; nastavni materijali mogu biti u obliku elektroničkih dokumenata, arhiva snimljenih predavanja (primjerice u formatu RealAudio, MPEG ili sličnom), a komunikacija može ići putem elektroničke pošte, foruma i sl.; u ovom obliku učenik tipično sam može odabrati kada želi učiti i kojim tempom (uz određene ograde),
- *suradno učenje* (engl. collaborative learning) podrazumijeva komunikaciju uživo (engl. online) i suradnju između članova grupe o specifičnoj temi; podrazumijeva uporabu razgovora (engl. chat), foruma i sličnih tehnologija u kojima može sudjelovati više sudionika istovremeno.

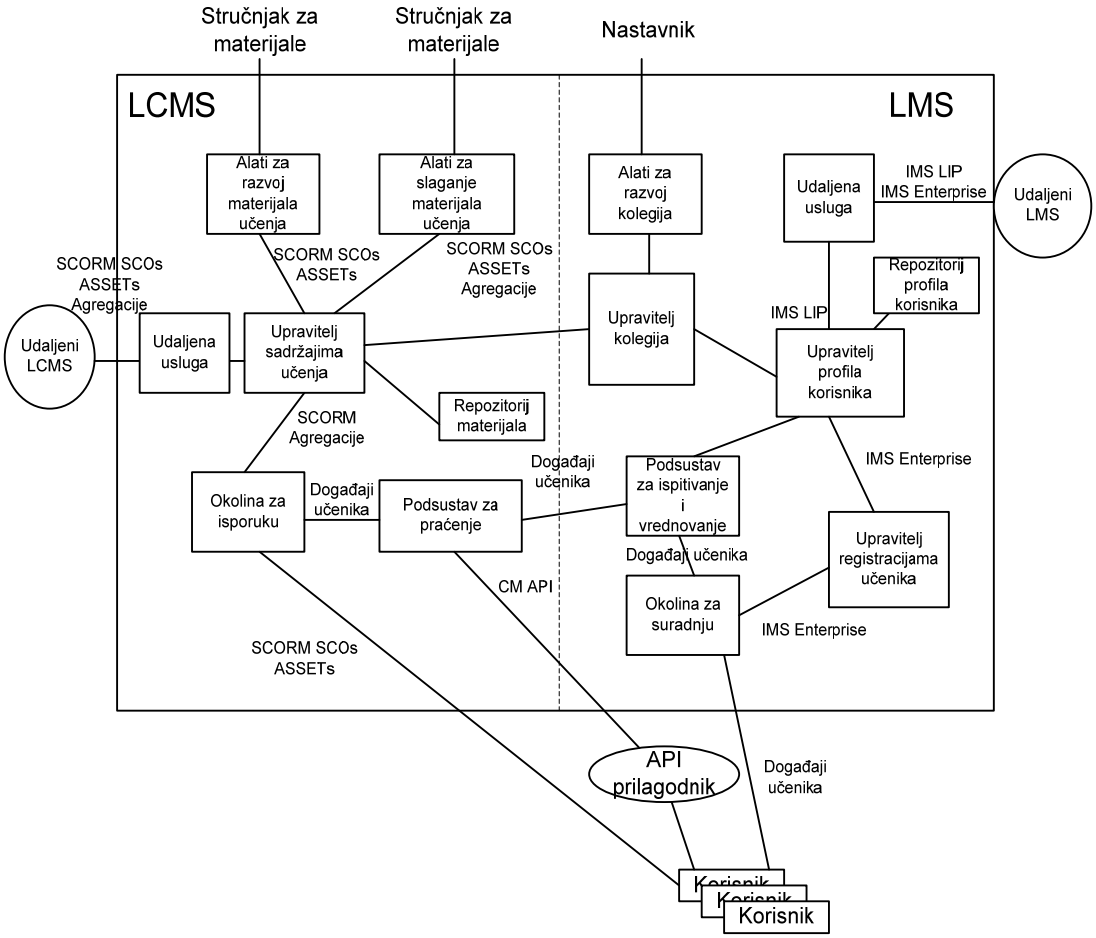


Slika 2-2. Klasifikacija sustava m-učenja

Jedan od novijih trendova u razvoju sustava e-učenja su sustavi m-učenja, čiji je zadatak približiti pogodnosti sustava e-učenja mobilnim korisnicima. Ovi bi sustavi trebali u potpunosti osigurati dostupnost sustava e-učenja sa svakog mjesta, u svakom trenutku i za svakoga. Opća podjela ovih sustava može se pronaći u [3] (vidi Slika 2-2).

Jedno od važnih pitanja koje treba razriješiti na zadovoljavajući način jest raznolikost karakteristika pristupnih terminala u sustavima m-učenja. Primjerice, dvije najvažnije karakteristike koje treba razmotriti su kvaliteta prikazne jedinice terminala (tekstovni zaslon ili grafički zaslon visoke rezolucije) te brzina pristupa sustavu m-

učenja (odnosno s obzirom na činjenicu da su sustavi m-učenja danas tipično poslužitelji temeljeni na tehnologijama Interneta, zanimljiva je brzina pristupa Internetu). Obzirom da ovi sustavi trebaju svima omogućiti učenje, potrebno je razraditi tehnike adaptacije obrazovnih sadržaja mogućnostima korisničkih terminala.



Slika 2-3. Funkcijski model sustava e-učenja prema [9]

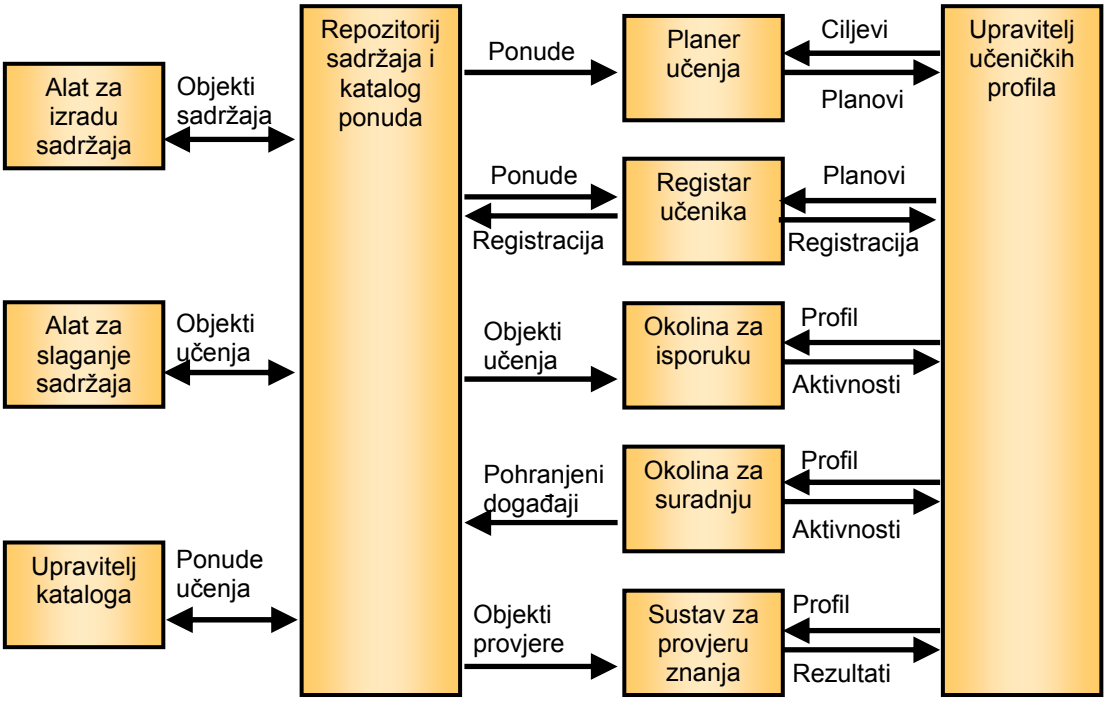
2.2 Modeli sustava e-učenja

Tijekom proteklih nekoliko dekada razvoja sustava e-učenja definiran je čitav niz vlasničkih modela i odgovarajućih implementacija (primjerice REF11, REF12). Međutim, promatra li se općeniti sustav e-učenja, može se analizirati funkcionalnost koju takav sustav mora sadržavati, čime se definira njegov *funkcijski model* (Slika 2-3, REF9).

Ovaj se model sastoji od dva dijela: LMS-a i LCMS-a. Sustav za upravljanje učenjem (LMS, engl. Learning Management System) dio je sustava zadužen za omogućavanje samog učenja te obuhvaća registraciju i rad s korisnicima, brigu oko korisničkih dozvola, stvaranje i oblikovanje kolegija, pružanje usluga koje omogućavaju suradnju korisnika te podsustav za testiranje odnosno provjeru znanja

korisnika (iako, obzirom na složenost ovog podsustava smatram da bi ovaj segment trebalo izdvojiti kao zasebnu zamjenjivu komponentu).

Sustav za upravljanje sadržajem učenja (LCMS, engl. Learning Content Management System) zadužen je za stvaranje, oblikovanje i upravljanje "sadržajima učenja". Ovaj dio omogućava stvaranje i višekratnu uporabu malih jedinica nastavnog materijala, grupiranje materijala u veće jedinice, uvoz i izvoz materijala iz sustava, čime se osigurava dijeljenje i razmjena stvorenog nastavnog materijala i višestruka iskoristivost. Zahvaljujući mogućnosti uvoza sadržaja standardiziranog formata omogućeno je i stvaranje nastavnog materijala vanjskim alatima.



Slika 2-4. Funkcijski model sustava e-učenja prema EduWorks

Naravno, radi ostvarivanja uspješne integracije ovih sustava (LMS-a i LCMS-a) potrebno je osigurati sučelje koje se temelji na otvorenim normama, te stoga osigurava međusobnu suradnju.

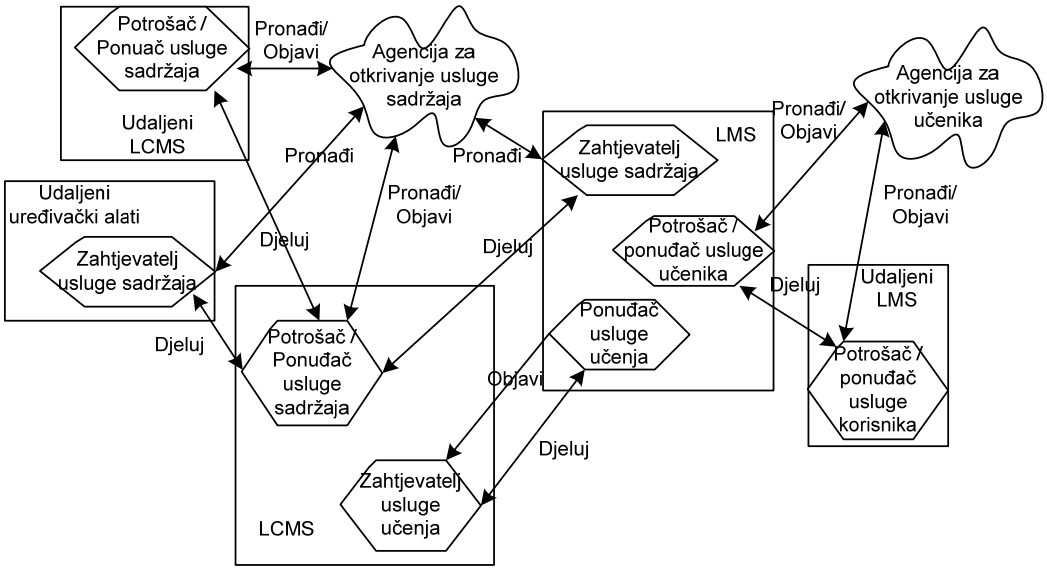
Slika 2-4 prikazuje još jedan pogled na funkcijski model sustava e-učenja, prema kompaniji EduWorks [4]. U ovom slučaju nema jasne identifikacije komponenti koje pripadaju LMS-u odnosno komponenti koje pripadaju LCMS-u. Međutim, pažljivom analizom ovaj se model može razložiti na tri dijela:

- LMS: čine ga komponente upravljač učeničkim profilima (engl. Learner Profile Manager), planer učenja (engl. Learning Planner), prijava učenika (engl. Learner Registration), okolina isporuke sadržaja (engl. Delivery Environment), repozitorij sadržaja i katalog ponuda (engl. Content Repository and Offering Catalog) te upravitelj katalozima (engl. Catalog Manager),
- LCMS: čine ga komponente okolina isporuke sadržaja (engl. Delivery Environment), repozitorij sadržaja i katalog ponuda (engl. Content

Repository and Offering Catalog), alati za izradu sadržaja (engl. Content Authoring Tools), te alati za slaganje sadržaja (engl. Content Assembly Tools), te

- CMS (engl. Course Management System): čine ga komponente upravljač učeničkim profilima (engl. Learner Profile Manager), okolina isporuke sadržaja (engl. Delivery Environment), okolina za suradnju (engl. Collaborative Environment), sustav za provjeru znanja (engl. Assessment/Testing Engine), alati za izradu sadržaja (engl. Content Authoring Tools) te alati za slaganje sadržaja (engl. Content Assembly Tools).

Pri tome je jasno vidljivo da se više komponenti dijeli između LMS-a, LCMS-a i CMS-a



Slika 2-5. Model sustava e-učenja temeljen na Web uslugama [9]

U današnje doba razvojem Interneta i potpornih tehnologija također je jasno izražen pomak prema novoj paradigmi izrade raspodijeljenih aplikacija – paradigmi temeljenoj na uslugama. U tom kontekstu, model sustava e-učenja temeljen na Web uslugama predložen je u [9] (Slika 2-5). Na slici se jasno mogu uočiti različiti dijelovi sustava e-učenja izvedeni kao zasebne komponente, koje međusobno komuniciraju uporabom Web usluga.

Razrada sustava e-učenja temeljenog na Web uslugama također se može pronaći u okviru magistarskog rada [21].

2.3 Norme u e-učenju

Iz dosadašnje rasprave jasno je kako je potpora učenju vrlo složen i zahtjevan proces, što rezultira kompleksnom građom sustava e-učenja. Također, danas sustavi e-učenja nisu monolitne zatvorene aplikacije već sustavi izgrađeni od manjih komponenti koje

međusobno surađuju. Kako bi se osiguralo da svi sastavni dijelovi sustava e-učenja mogu međusobno surađivati, potrebno je osigurati određeno dijeljeno znanje i komuniciraju preko dobro definiranih sučelja, što je uloga normi. Norme u e-učenju mogu se podijeliti u pet kategorija [9]:

- meta-podaci
- pakiranje sadržaja učenja
- profil učenika
- registracija učenika
- komunikacija sadržaja učenja

Kako bi se podržalo pohranjivanje, indeksiranje i pretraživanje te dohvat sadržaja učenja (engl. learning content) nužno je osigurati zajednički jezik za opisivanje tih sadržaja i njihovih kataloga. Time se ujedno želi osigurati i ispravan rad niza alata (koji razumiju korišteni jezik) te obavljanje operacija nad više repozitorija sadržaja učenja. Dvije najznačajnije norme na ovom području su *Learning Object Metadata – LOM* tijela *IEEE Learning Technology Standards* [13], te *Dublin Core Metadata* [14].

Kako bi se omogućio izvoz nastavnog materijala i složenih kolegija, nužno je osigurati specifikacije za pakiranje sadržaja. Tri najznačajnije specifikacije na ovom području su *The IMS Content Packaging Specification* [15], *The IMS Simple Sequencing Specification* [15] te *ADL Sharable Content Object Reference Model (SCORM)* [16].

Najvažnija specifikacija za pohranu profila učenika jest *IMS Learner Information Package (LIP)* [15]. Profil učenika sadrži učenikove osobne podatke, planove i povijest učenja, certifikate, procjene trenutnog učenikovog znanja i sl.

Registracija učenika sadrži informacije o nastavnim sadržajima (tj. kolegijima) koje učeniku treba staviti na raspolaganje, prilikom dolaska na novi sustav. Dvije najznačajnije specifikacije su *IMS Enterprise Specification* [15] te *Schools Interoperability Framework* [17].

Komunikacija sadržaja učenja obuhvaća komunikaciju između samog sadržaja u trenutku pokretanja istoga, i sustava koji ugošćuje dotični sadržaj. Svrha ove komunikacije jest razmjena podataka o samom učeniku, te dostava podataka o učenikovim prethodnim aktivnostima. Najvažnija specifikacija na ovom području je *ADL Sharable Content Object Reference Model (SCORM)* [16].

Na području e-ispitivanja koje je u funkcijskim modelima sustava e-učenja pokriveno podsustavima za testiranje tj. provjeru znanja također je potrebno spomenuti jednu od trenutno najvažnijih specifikacija: *IMS Question and Test Interoperability* [18], koja definira način za opisivanje pitanja za provjeru znanja.

U nastavku će najprije biti ukratko opisani LOM i SCORM, a potom slijedi opis specifikacije *IMS Question and Test Interoperability*.

2.4 Norma LOM

LOM [20] je norma za opisivanje objekata učenja temeljen na projektima ARIADNE [19] i IMS [15], te na rezultatima rada grupe Dublin Core [14], a rezultat je rada radne grupe unutar *The IEEE Learning Technology Standards Committee (LTSC)*. Namjena norme jest omogućiti sustavima za učenje rukovanje, pretraživanje, ocjenjivanje i razmjenu objekata učenja. Pri tome norma ne propisuje način na koji će sustavi interno pohranjivati ove podatke. U kontekstu ove norme objekt učenja (engl. learning object) jest bilo koji entitet (digitalni ili ne-digitalni) koji se može koristiti za potrebe učenja, obrazovanja ili vježbanja.

U okviru LOM-a, metapodaci služe za opis bitnih karakteristika samih objekata učenja. Prema osnovnoj shemi LOM verzije 1.0 podaci su grupirani u 9 kategorija (Tablica 2-1).

Tablica 2-1. Kategorije podataka osnovne sheme LOM verzije 1.0

Naziv kategorije	Opis
općenito (engl. General)	kategorija općih informacija koje opisuju objekt učenja kao cjelinu
životni ciklus (engl. Lifecycle)	kategorija za informacije koje se odnose na povijest i trenutno stanje objekta učenja (primjerice, uključuje i popis sudionika koji su utjecali na objekt učenja tijekom njegove evolucije)
meta-Metapodaci (engl. Meta-Metadata)	kategorija za informacije o samim metapodacima (dakle ne o objektu učenja, već o metapodacima koji ga opisuju)
tehničko (engl. Technical)	kategorija tehničkih zahtjeva i karakteristika objekta učenja
obrazovno (engl. Educational)	kategorija obrazovnih i pedagoških karakteristika objekta učenja
prava (engl. Rights)	kategorija za opis intelektualnih prava i uvjeta pod kojima se objekt učenja smije koristiti
odnosi (engl. Relation)	kategorija u kojoj se nalaze informacije o odnosima između objekta učenja i drugih povezanih objekata učenja
opisi (engl. Annotation)	kategorija za napomene i komentare o načinu uporabe objekta učenja u obrazovnom procesu (ujedno sadrži i informacije o autorima i vremenu stvaranja samih komentara)
klasifikacija (engl. Classification)	opis samog objekta učenja pomoću odgovarajućeg klasifikacijskog sustava (pri tome nije unaprijed propisano koji sustav koristiti, što znači da se time sam LOM može dodatno proširiti)

Imena pojedinih kategorija prilikom uporabe moraju se koristiti u engleskom izvorniku (što je u tablici prikazano u zagradama), jer ona ujedno predstavljaju i nazive koji će ulaziti u imena samih svojstava prilikom opisa.

Tablica 2-2. Podaci definirani u okviru kategorije General norme LOM

Broj	Ime	Opis
1	općenito (engl. General)	kategorija grupira informacije opće informacije o objektu učenja
1.1	identifikator (engl. Identifier)	globalno jedinstven identifikator objekta učenja
1.1.1	katalog (engl. Catalog)	ime ili oznaka kataloga prema kojem se tvori identifikator
1.1.2	element (engl. Entry)	vrijednost identifikatora koji jednoznačno određuje objekt učenja
1.2	naziv (engl. Title)	ime dodijeljeno objektu učenja
1.3	jezik (engl. Language)	primarni jezik koji se koristi prilikom komunikacije s čovjekom u objektu učenja
1.4	opis (engl. Description)	tekstovni opis sadržaja objekta učenja
1.5	ključne riječi (engl. Keyword)	ključna riječ ili fraza koja opisuje temu objekta učenja
1.6	pokrivanje (engl. Coverage)	vrijeme, kultura ili zemljopisno područje na koje se odnosi objekt učenja
1.7	struktura (engl. Structure)	kako je sam objekt učenja organiziran / strukturiran
1.8	stupanj agregacije (engl. Aggregation Level)	funkcijska zrnatost samog objekta učenja

Upravo nabrojane kategorije služe za grupiranje podataka (engl. data elements) kojima se opisuju objekti učenja. Model podataka LOM-a jest hijerarhijski te uključuje agregacijske podatke (engl. aggregate data elements) te jednostavne podatke (engl. simple data elements). Za svaki podatak osnovna shema LOM definira:

- *ime* – ime po kojem se podatak referira,
- *objašnjenje* – definicija samog podatka,
- *veličinu* – broj dozvoljenih vrijednosti,
- *poredak* – je li poredak vrijednosti važan ili nije (odnosi se na podatke koji imaju liste vrijednosti), te

- *primjer* – ilustrativni primjer koji demonstrira način uporabe samog podatka.

Za jednostavne podatke osnovna shema LOM također definira:

- *prostor vrijednosti* – skup dozvoljenih vrijednosti koje podatak može poprimiti (obično u obliku rječnika ili reference na neku drugu normu), te
- *tip podatka* – govori kakvog je tipa podatak. Za podatke je definirano 6 mogućih tipova: LangString, DateTime, Duration, Vocabulary, CharacterString ili Undefined.

Primjer podataka definiranih ovom normom prikazuje Tablica 2-2.

Tablica 2-2 prikazuje samo manji podskup podataka definiranih normom LOM (točnije, kategoriju *General*), pri čemu prvi stupac opisuje hijerarhijski položaj samog podatka. Tako je podatak *General* zapravo agregacija 8 podataka: *General.Identifier*, *General.Title*, *General.Language*, *General.Description*, *General.Keyword*, *General.Coverage*, *General.Structure* te *General.Aggregation Level*. *General.Identifier* po tipu je nova agregacija, i sastoji se od podataka *General.Identifier.Catalog* i *General.Identifier.Entry*. Na sličan način definirano je i preostalih 8 kategorija, koje zbog obimnosti same norme ovdje neće biti prenesene. Tablica 2-2 također ne sadrži sve elemente definicije podataka koje propisuje LOM osnovna shema, no zainteresiran čitatelj može ih pronaći u [20].

2.5 Norma SCORM

Sharable Content Object Reference Model (SCORM) [16] je referentni model koji specificira jedinice materijala za učenje (engl. learning content) i način njihovog opisivanja, pohranu te način prikaza u raspodijeljenim sustavima e-učenja. Pri tome SCORM nije u potpunosti nova norma, već predstavljam model koji se temelji na skupu postojećih normi i specifikacija razvijenih kroz niz prethodnih projekata. SCORM se sastoji iz tri specifikacije: "Model agregiranja sadržaja" (engl. "Content Aggregation Model") i "Izvršna okolina" (engl. "Run-Time Environment") za jedinice materijala za učenje podržavaju pružanje adaptivnih instrukcija temeljenih na učenikovim ciljevima, sklonostima i performansama (te drugim faktorima) dok "Uslijeđivanje i kretanje" (engl. "Sequencing and Navigation") osigurava dinamički prikaz jedinica materijala za učenje temeljeći se na potrebama učenika.

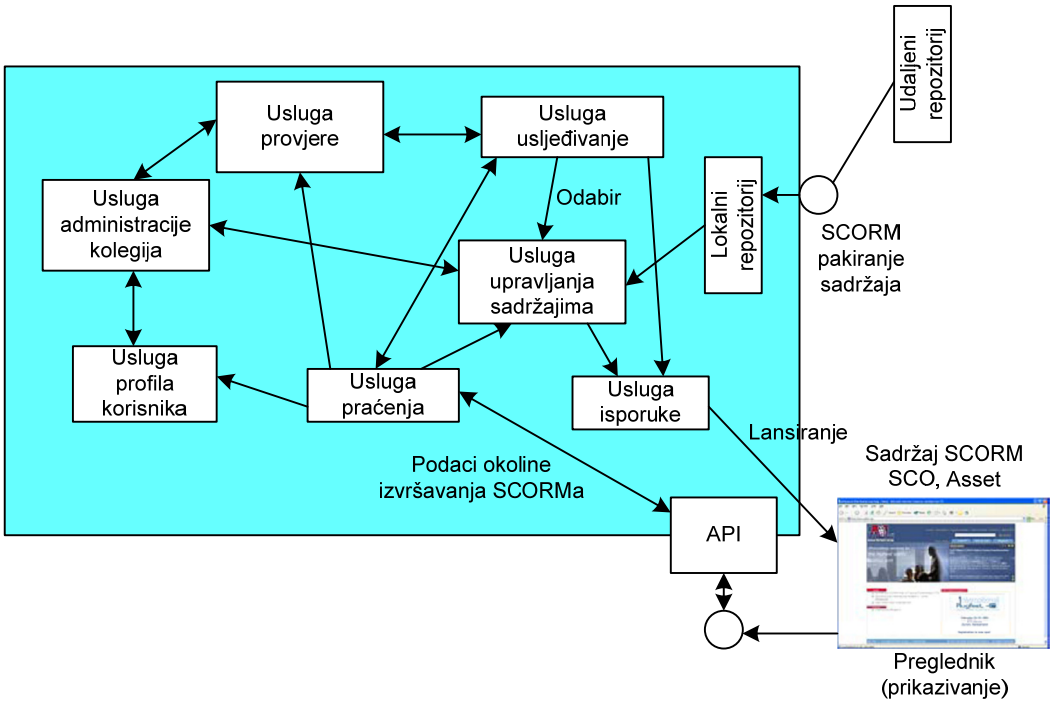
Pri tome su kao polazna točka za razvoj SCORM-a uzeta tri kriterija:

- SCORM mora ponuditi jasne smjernice koje će biti razumljive razvijateljima jedinica materijala za učenje,
- želja je da se SCORM prihvatiti, razumije i koristi u što je moguće široj zajednici korisnika, te
- SCORM mora omogućiti postojećim modelima vođenja učenika laganu implementaciju temeljenu na SCORM-u. Drugim riječima, SCORM ne želi definirati kako će učenik učiti, već želi omogućiti postojećim i razrađenim sustavima učenja i poučavanja uporabu postojećih algoritama, pri čemu se sadržajima pristupa kroz prizmu SCORM-a.

Sam SCORM utemeljen je na nekoliko nefunkcijskih zahtjeva kojima se vode i sve promjene i dodaci u model:

- *pristupivost* (engl. accessibility) – mogućnost pronalaženja i pristupa instrukcijskim komponentama s udaljenih lokacija i dostava istih na mnoštvo drugih lokacija,
- *prilagodivost* (engl. adaptability) – mogućnost prilagođavanja načina poučavanja potrebama individua i organizacija,
- *priuštivost* (engl. affordability) – mogućnost povećanja efikasnosti i produktivnosti smanjenjem vremena i troškova povezanih s obavljanjem samog procesa učenja i poučavanja,
- *trajnost* (engl. durability) – sposobnost otpornosti na tehnološki razvoj i promjene (odnosno, razvojem tehnologije neće biti potrebe za skupim prilagodbama jedinica materijala učenja, što bi uključivalo novi dizajn, ponovno konfiguriranje te novo kodiranje),
- *mogućnost suradnje* (engl. interoperability) – sposobnost prijenosa jedinica materijala učenja s jedne lokacije (i jednog sustava) na drugu lokaciju (i u drugi sustav) gdje se koristi neki drugi skup potpornih alata,
- *višestruka iskoristivost* (engl. reusability) – mogućnost ugradnje jedinica materijala za učenje u više aplikacija i konteksta.

Osim ovoga, osnovna pretpostavka norme SCORM je utemeljenost na tehnologijama Weba.



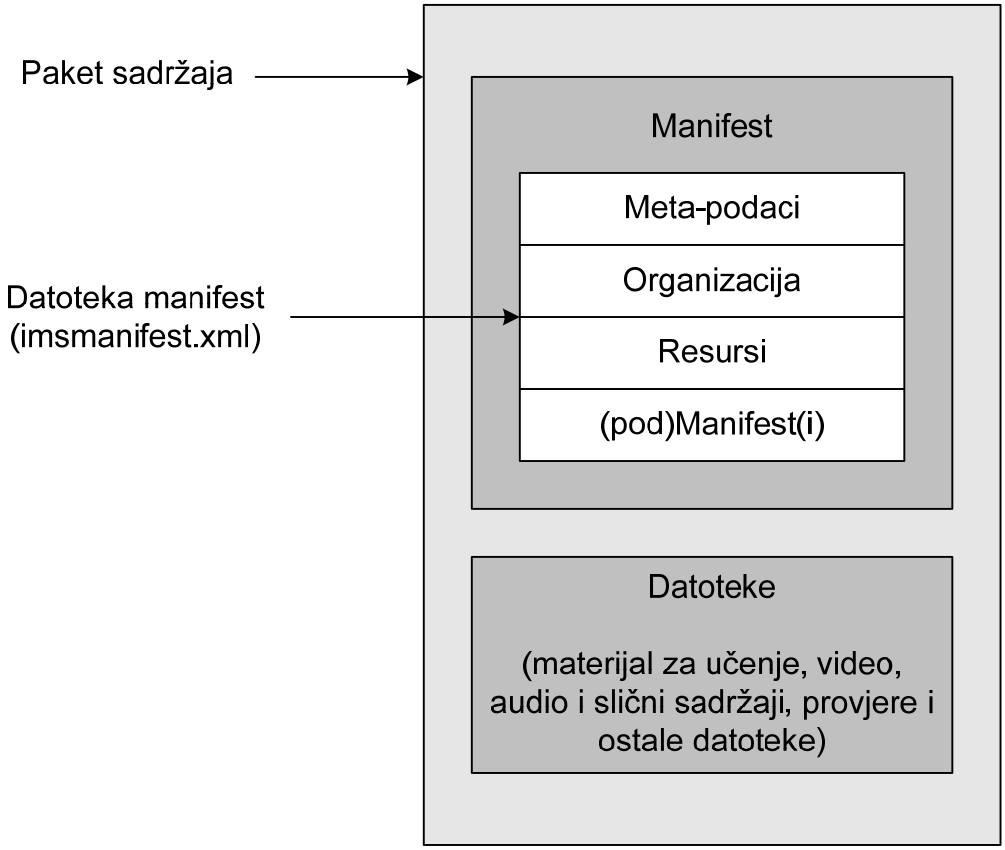
Slika 2-6. Model LMS sustava prema normi SCORM

Slika 2-6 prikazuje model LMS prema viđenju specifikacije SCORM. Na slici su također prikazana i mjesta gdje specifikacija SCORM dolazi do izražaja:

- razmjena jedinica materijala za učenje (na slici je vidljiv uvoz ovih jedinica iz udaljenog repozitorija),

- isporuka korisniku jedinica materijala za učenje kroz proces pokretanja jedinica, te
- komunikacija jedinice materijala za učenje sa samim sustavom učenja, u svrhu dohvata/pohrane dinamičkih podataka vezanih uz korisnika kojemu je jedinica prikazana i samu jedinicu materijala za učenje.

Slika također jasno ilustrira da se zahvaljujući normi SCORM i specifikaciji za pakiranje sadržaja te njihovog opisivanja sadržaji učenja mogu pronalaziti/dohvaćati iz udaljenih repozitorija potom koristiti u lokalnom sustavu.



Slika 2-7. Konceptualni pogled na paket jedinice materijala za učenje

Specifikacija SCORM sastoji se od 4 dijela:

- *pregled* (engl. *overview*) – sadrži opće konceptualne informacije o SCORM-u i uvodi niz elemenata ali na visokoj konceptualnoj razini,
- *model agregiranja sadržaja* (engl. *Content Aggregation Model – CAM*) – pokriva grupiranje, opisivanje i pakiranje jedinica materijala za učenje; definira pojmove: SCO, Asset, Content Aggregation, Package, Package Interchange File (PIF), Meta-data, Manifest, Sequencing Information te Navigation Information,
- *izvršna okolina* (engl. *Run-Time Environment – RTE*) – opisuje pokretanje jedinica materijala za učenje, komunikaciju s LMS-om te praćenje, prijenos podataka i obradu pogrešaka; definira pojmove: API, API Instance, Launch,

Session Methods, Data Transfer Methods, Support Methods, Temporal Model, Run-Time Data Model,

- *uslijeđivanje i kretanje* (engl. *Sequencing and Navigation – SN*) – opisuje redoslijed i upravljanje kretanjem kroz jedinicu materijala za učenje; definira pojmove: Activity Tree, Learning Activities, Sequencing Information, Navigation Information, Navigation Data Model.

Jedinica materijala za učenje može se sastojati od jedne ili više datoteka. Zbog toga SCORM definira način na koji se sve datoteke jedinice materijala za učenje mogu složiti u jedan paket (Slika 2-7). Sa slike je jasno vidljivo da jedan paket, uz same datoteke koje čine materijale učenja sadrži i posebnu datoteku (manifest) koja opisuje sadržaj paketa. Ta datoteka sadrži informacije definirane jezikom koji je razumljiv svim sustavima kompatibilnim s normom SCORM.

Sa stanovišta raspodijeljenih i otvorenih sustava e-učenja, SCORM je specifikacija koja ima svijetlu budućnost. Osnovna ideja koja definira elementarne jedinice za učenje i postupa s njima kao sa samostalnim objektima, pri čemu definira način za komunikaciju između samog objekta i sustava e-učenja jest definitivno pravi pristup, koji nudi veliku slobodu i ekspresivnost u radu s istima (što je već davno prije viđen pristup u drugim područjima; primjerice tehnologija Servlet [26]). Međutim, sa stanovišta e-ispitivanje, treba primijetiti da niti LOM niti SCORM nisu zamišljeni za opisivanje/rad sa zadacima za provjeru znanja, već isključivo za upravljanje pojavnim oblicima materijala za učenje (objekti učenja, jedinice materijala za učenje). Stoga će u nastavku pažnja biti posvećena postojećoj specifikaciji za rad sa zadacima.

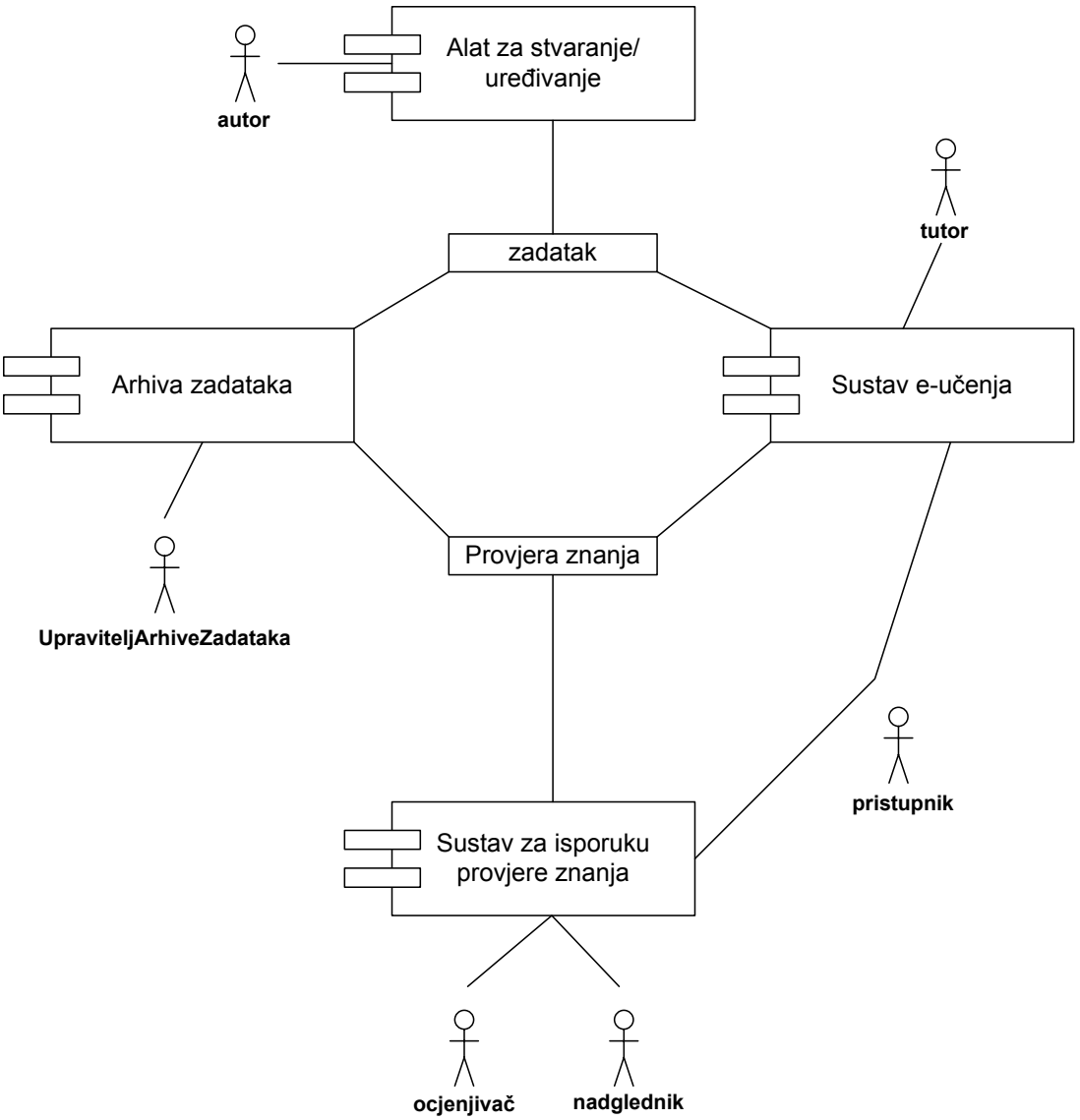
2.6 Norma "IMS Question and Test Interoperability"

IMS Global Learning Consortium [15] definirao je specifikaciju *IMS Question and Test Interoperability (QTI)* koja je trenutno u konačnoj drugoj verziji. Ova specifikacija opisuje model podataka za prikaz pitanja (engl. assessmentItem), provjere znanja (engl. assessment) te rezultata provjere znanja korisnika. Namjena specifikacije je također osigurati razmjenu opisa i rezultata provjere znanja između različitih alata i time uspostaviti okruženje u kojem alati različitih proizvođača te različiti sustavi za provjere znanja međusobno surađuju.

Koncepti na kojima se specifikacija temelji definirani su na visokoj apstraktnoj razini – međutim, definiran je i način iskazivanja tih koncepata uporabom jezika XML što je i preporučeni način.

Prvi nacrt specifikacije pojavio se je u ožujku 1999. godine, a verzija 1.0 u svibnju 2000. godine. Trenutno je aktualna verzija 2.0 objavljena u siječnju 2005. godine. Prema navodima u pregledu *IMS Question and Test Interoperability Overview* [22], verzija 1.x do drugog mjeseca 2002. godine skinuta je preko 6000 puta, što jasno ukazuje na velik interes.

QTI definira nekoliko sustava i odgovarajućih aktora (UML terminologijom; Slika 2-8).



Slika 2-8. QTI pogled na proces provjere znanja. UML notacija.

Pri tome su definirani sustavi:

- *alat za stvaranje/uređivanja* – aplikacije koji služe za izradu zadataka,
- *arhiva zadataka* – repozitorij u koji se pohranjuju svi zadaci stvoreni prethodnim alatom,
- *sustav za isporuku provjere znanja* – podsustav koji je zadužen za proces isporuke provjere znanja učeniku; sustav sadrži potporu za automatsko vrednovanje zadataka (ukoliko to sami zadaci dopuštaju), ili može učenikove odgovore slati ocjenjivaču (živoj osobi) na ocjenjivanje,
- *sustav e-učenja* – sustav kroz koji učenik pristupa učenju, i koji određuje što i kojim redoslijedom učenik treba dobiti (možda uz suradnju tutora).

Aktori definirani prethodnim modelom su:

- *autor* – osoba koja definira zadatak; to u jednostavnim slučajevima može biti jedna osoba, ili u složenijim slučajevima čitav tim,
- *upravitelj arhive zadataka* – osoba koja je zadužena za održavanje arhive zadataka,
- *nadglednik* – osoba koja je zadužena za nadgledanje procesa isporuke provjere znanja; QTI nadglednika definira kao bilo koju osobu (osim pristupnika) koja je uključena u proces isporuke provjere znanja, ali ne sudjeluje u vrednovanju pristupnikovih odgovora,
- *ocjenjivač* – osoba (ili vanjski sustav) zadužena za vrednovanje korisnikovih odgovora na pitanja u provjeri znanja; ova uloga često nije potrebna jer će

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Thie example adapted from the PET Handbook, copyright University of Cambridge ESOL
Examinations -->
<assessmentItem xmlns="http://www.imsglobal.org/xsd/imsqti_v2p0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p0 imsqti_v2p0.xsd"
   identifier="choice" title="Unattended Luggage" adaptive="false" timeDependent="false">
   <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="identifier">
      <correctResponse>
         <value>ChoiceA</value>
      </correctResponse>
   </responseDeclaration>
   <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="integer">
      <defaultValue>
         <value>0</value>
      </defaultValue>
   </outcomeDeclaration>
   <itemBody>
      <p>Look at the text in the picture.</p>
      <p>
         <img src="images/sign.png" alt="NEVER LEAVE LUGGAGE UNATTENDED"/>
      </p>
      <choiceInteraction responseIdentifier="RESPONSE" shuffle="false" maxChoices="1">
         <prompt>What does it say?</prompt>
         <simpleChoice identifier="ChoiceA">
               You must stay with your luggage at all times.
         </simpleChoice>
         <simpleChoice identifier="ChoiceB">
            Do not let someone else look after your luggage.
         </simpleChoice>
         <simpleChoice identifier="ChoiceC">
            Remember your luggage when you leave.
         </simpleChoice>
      </choiceInteraction>
   </itemBody>
   <responseProcessing
      template="http://www.imsglobal.org/question/qti_v2p0/rptemplates/match_correct"/>
</assessmentItem>
```

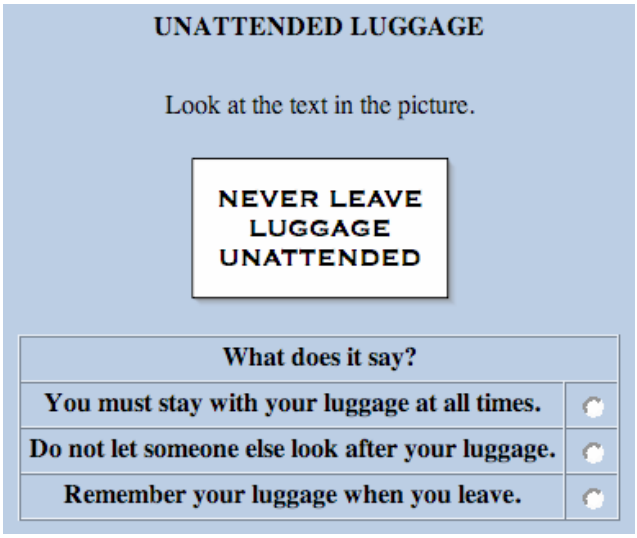
Slika 2-9. Jednostavna ABC-pitalica

jednostavni zadaci imati ugrađenu mogućnost automatskog ocjenjivanja,

- *tutor* – osoba uključena u upravljanje i potporu procesa učenja,
- *pristupnik* – osoba koja je ispitivana kroz provjeru znanja.

U nastavku slijedi nekoliko zadataka opisanih u skladu s QTI. Ti su primjeri preuzeti iz dokumenta [23].

Prvi primjer prikazuje jednostavnu ABC pitalicu s jednim točnim odgovorom, koji je



Slika 2-10. Prikaz jednostavne ABC-pitalice

u skladu s normom QTI opisan u formatu XML (Slika 2-9).

Prikazani primjer definira pitanje kroz četiri dijela: odgovor (oznaka responseDeclaration) koji može biti samo jedan, način bodovanja (oznaka outcomeDeclaration), tijelo pitanja (oznaka itemBody) koje se u ovom slučaju sastoji od pitanja te popisa ponuđenih odgovora, te način vrednovanja rješenja (oznaka responseProcessing).

Slika 2-10 ilustrira jedan mogući prikaz ovog pitanja na zaslonu pristupnika.

Drugi primjer je zadatak u kojem je na slici potrebno odabrati lokaciju na slici. Konkretno, na karti Engleske potrebno je odabrati položaj grada Edinburgha. Slika 2-12 prikazuje XML dokument s definicijom pitanja.

Primjer je strukturno jednak prethodnom (sastoji se od 4 dijela) pri čemu je način odgovaranja definiran povezivanjem lokacije na koju pristupnik klikne i odgovora ("0" ili "1"), a pitanje sadrži sliku.

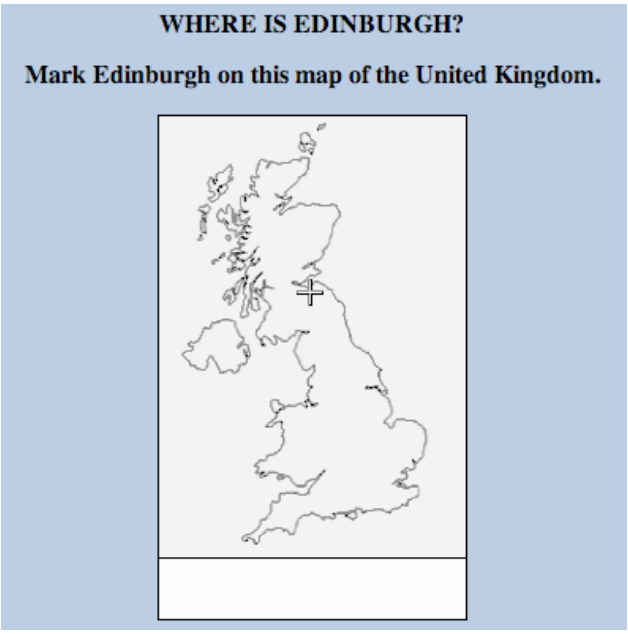
Slika 2-11 ilustrira jedan mogući prikaz ovog pitanja na zaslonu pristupnika.

Ideja QTI specifikacije jest opis zadataka i svega što je uz to potrebno jezikom XML, na standardizirani način. U osnovi, to je pokušaj da se osigura nezavisan razvoj zadataka s jedne strane i njihove uporabe u različitim sustavima s druge strane – čime se, jasno, automatski mora osigurati mogućnost njihove međusobne suradnje.

```
<?xml version="1.0" encoding="UTF-8"?>
<assessmentItem xmlns="http://www.imsglobal.org/xsd/imsqti_v2p0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p0 imsqti_v2p0.xsd"
   identifier="selectPoint" title="Where is Edinburgh?" adaptive="false" timeDependent="false">
   <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="point">
      <correctResponse>
         <value>102 113</value>
      </correctResponse>
      <areaMapping defaultValue="0">
         <areaMapEntry shape="circle" coords="102,113,8" mappedValue="1"/>
      </areaMapping>
   </responseDeclaration>
   <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="integer"/>
   <itemBody>
      <selectPointInteraction responseIdentifier="RESPONSE" maxChoices="1">
         <prompt>Mark Edinburgh on this map of the United Kingdom.</prompt>
         <object type="image/png" width="196" height="280" data="images/uk.png">
            UK Map</object>
      </selectPointInteraction>
   </itemBody>
   <responseProcessing
template="http://www.imsglobal.org/question/qti_v2p0/rptemplates/map_response_point"/>
</assessmentItem>
```

Slika 2-12. Zadatak oblika "odaberi točku"

Međutim, sa stanovišta ekspresivnosti ovog pristupa, vrlo se brzo dolazi do ograničenja. Naime, osim krajnje jednostavnih zadataka u kojima će se malo stvari mijenjati, za kvalitetnu provjeru znanja potrebno je osigurati potporu za složene procedure odabira parametara zadatka prilikom stvaranja pitanja za pristupnika, potporu za prikaz pitanja s kompleksnim korisničkim sučeljem, potporu za složene tehnike vrednovanja pristupnikovih odgovora, te mogućnost izolacije komponenti koje su korištene u više zadataka u zasebne komponente koje će biti na raspolaganju



Slika 2-11. Odabir točke na slici

u trenutku stvaranja/vrednovanja zadatka. QTI nažalost ne nudi odgovarajuću podršku za navedene probleme.

3 E-ispitivanje u postojećim sustavima

Jedna od temeljnih funkcija koje sustavi e-učenja moraju ponuditi jest provjera usvojenog znanja. U ovom poglavlju opisat će se ukratko mogućnosti dva danas najpoznatija sustava koji nude mogućnosti e-ispitivanja, te se standardno nude na uporabu kolegijima Fakulteta elektrotehnike i računarstava: WebCT i MOODLE.

3.1 Osvrt na mogućnosti sustava WebCT

WebCT sustav je sustav za potporu e-učenju te u toj domeni ima bogat skup mogućnosti. Jedna od mogućnosti jest i ispitivanje korisnika uporabom kvizova. Ova posljednja mogućnost ukratko se analizira u nastavku. Na razini samog kviza, sustav WebCT nudi sljedeće mogućnosti:

- način isporuke pitanja (sva odjednom, jedno po jedno s mogućnošću povratka te jedno po jedno bez mogućnosti povratka),
- trajanje kviza u minutama, satima ili danima,
- broj pokušaja rješavanja kviza (1, 2, 3, 4, 5 ili neograničeno),
- minimalno vrijeme između ponovljenih pokušaja rješavanja kviza (u minutama, satima ili danima),
- vrijeme otvaranja kviza,
- vrijeme zatvaranja kviza,
- selektivno dopuštanje pristupa kvizu samo određenim studentima,
- zaporka za početak pisanja testa,
- dozvoljene IP adrese (samo jedna maska oblika X.Y.Z.W),
- izračun ocjene kviza za slučaj da se može polagati iz više pokušaja – tada se može uzeti ocjena prvog kviza ili ocjena zadnjeg kviza, srednja ocjena svih kvizova ili pak najveća ocjena kviza,
- kada studenti mogu vidjeti rezultate (kombinacije uvjeta "nakon što je kviz poslan" i "nakon što je kviz ocijenjen", odnosno po isteku vremenskog ograničenja),
- što točno od rezultata student smije vidjeti.

Od raspoloživih tipova pitanja, WebCT nudi 5 tipova:

- *jednostruki-višestruki odgovori*; može se definirati naziv pitanja, tekst pitanja, je li zadatak s jednim točnim odgovorom ili s više njih, je li bodovanje kumulativno ili "sve-ili-ništa", ima li negativnih bodova, treba li opcije ispremiješati posredstvom slučajnog mehanizma te niz opcija koje će biti ponuđene studentu,
- *uparivanje*; može se definirati naziv pitanja, tekst pitanja, način bodovanja odgovora (nosi li svaki ispravan par određen broj bodova, ili je potrebno ispravno označiti sve parove kako bi se dobili bodovi) te niz od do pet parova koje student dobiva na uparivanje,

- *izračunski zadatak*; može se definirati naziv pitanja, tekst pitanja, način bodovanja odgovora, formula po kojoj će se izračunati točan odgovor (uz elementaran skup podržanih funkcija) te mogućnost dodavanja mjernih jedinica u odgovor,
- *kratak odgovor*; može se definirati naziv pitanja, tekst pitanja te popis mogućih odgovora (koji se mogu prepoznavati uporabom regularnih izraza),
- *dugi tekst*; može se definirati naziv pitanja, tekst pitanja, tekst koji će biti upisan kao početni odgovor te tekst koji predstavlja točan odgovor.

Kao što se iz ove kratke analize može vidjeti, nisu podržani dinamički zadaci u kojima se pitanja stvaraju po složenim algoritmima, te ocjenjivanje istih.

3.2 Osvrt na mogućnosti sustava MOODLE

Sustav MOODLE nije isključivo namijenjen samo e-ispitivanju, već je, baš kao i WebCT sustav za potporu e-učenju. Međutim, ograničimo li se na njegove mogućnosti vezane uz definiranje kvizova (što je implementirana potpora e-ispitivanju), sustav MOODLE nudi niz parametara koji se mogu podešavati na dvije razine. Opcije koje se nude na razini samog kviza su:

- vrijeme otvaranja kviza,
- vrijeme zatvaranja kviza,
- vremensko ograničenje (opcionalno) od 1 do 230 minuta. U slučaju detekcije prevare kviz se boduje s 0 bodova,
- broj pitanja po stranici. Mogućnosti su 1 do 50 te neograničeno,
- miješanje redoslijeda pitanja posredstvom slučajnog mehanizma,
- miješanje redoslijeda ponuđenih odgovora posredstvom slučajnog mehanizma,
- broj dozvoljenih pokušaja rješavanja kviza. Mogućnosti su 1 do 6 te neograničeno,
- gradi li se novi pokušaj rješavanja kviza na prethodnom pokušaju ili ne,
- metoda ocjene kviza. Mogućnosti su: najviša ocjena, srednja ocjena, ocjena prvog pokušaja, ocjena zadnjeg pokušaja,
- adaptivni način rada. U adaptivnom načinu rada ukoliko učenik odgovori pogrešno na pitanje, pitanje će mu se ponoviti. Međutim, svako ponavljanje umanjuje bodove za faktor kazne,
- da li se primjenjuje faktor kazne,
- broj decimalnih mjesta koja će biti prikazana u ocjeni kviza,
- kada student može vidjeti svoje prethodne odgovore/bodove/povratne informacije – odmah i/ili naknadno, ali dok je kviz još otvoren i/ili nakon zatvaranja kviza,
- prikaz kviza u sigurnom prozoru,

- zaporka za početak pisanja kviza,
- IP adrese s kojih se može pristupiti kvizu,
- način grupiranja,
- je li kviz vidljiv studentima.

Jednom kada je kviz definiran, može se krenuti na odabir pitanja koja će biti uključena u kviz (ili stvaranje istih, ukoliko prethodno već nisu stvorena). Od podržanih tipova pitanja, MOODLE nudi selekciju od 9 tipova:

- *odabir višestrukih odgovora*; pitanje se sastoji od imena, teksta pitanja i ponuđenih odgovora; za svaki ponuđeni odgovor definira se i povratna poruka te točnost odgovora (u opsegu 100% do -100%), te faktor kazne; također, može se podesiti je li samo jedan odgovor točan, ili više odgovora može biti točno,
- *točno/netočno*; pitanje se sastoji od imena, teksta pitanja i indikacije točnosti tvrdnje; dodatno se mogu još definirati faktor kazne, povratna poruka za slučaj da je student odgovorio "točno" te povratna poruka za slučaj da je student odgovorio "netočno",
- *kratki odgovor*; pitanje se sastoji od imena, teksta pitanja te liste mogućih odgovora i pripadnim stupnjem točnosti; usporedba s odgovorima može biti osjetljiva na velika i mala slova, ili neosjetljiva,
- *numeričko pitanje*; pitanje se sastoji od imena, teksta pitanja, točne brojčane vrijednosti odgovora, numeričke tolerancije za priznavanje studentovog odgovora, opcionalne mjerne jedinice te skupa parova (mjerna jedinica, množitelj) kako bi se studentov odgovor mogao usporediti s točnim odgovorom,
- *izračunsko pitanje*; pitanje se sastoji od imena, teksta pitanja koje može sadržavati varijable, formule po kojoj se računa točan odgovor, faktora kazne, mjernih jedinica te tolerancije unutar koje student može pogriješiti u izračunu, a da mu se prizna točan odgovor; od podržanih operacija tu su zbrajanje, oduzimanje, množenje, dijeljenje, modulo operacija te neke elementarne funkcije (poput sinusa i kosinusa); za svaku od korištenih varijabli može se definirati iz kojeg će se skupa vrijednosti birati vrijednost,
- *uparivanje*; pitanje se sastoji od imena i općenitog teksta pitanja nakon kojeg slijedi niz potpitanja s definiranim točnim odgovorom,
- *opisivanje*; pitanje se sastoji od imena i teksta pitanja,
- *slučajno uparivanje kratkih odgovora*; pitanje se sastoji od imena, teksta pitanja te izmjenjivog broja kratkih pitanja koja će se uzeti i studentu prikazati kao skup pitanja i skup odgovarajućih odgovora, a student će trebati upariti pitanja s odgovorima,
- *ugrađeni odgovori (format Cloze)*; pitanje se sastoji od imena te teksta pitanja prema posebnom formatu, gdje se kroz tekst određenim tekstovnim oznakama ubacuju različite vrste pitanja; trenutna verzija MOODLE-a zbog

složenosti ovog formata još nema grafičko sučelje za uređivanje ove vrste pitanja.

3.3 Kratka analiza

Nakon pomnijeg proučavanja sustava WebCT i MOODLE nameće se nekoliko zaključaka:

- e-ispitivanje nije središnja i važna tema, već samo jedna od mogućnosti koje ovi sustavi nude riješeno u skladu s takvim stavom,
- oba sustava nude dobro razrađeno korisničko sučelje i izvrsne mogućnosti rada sa statičkim vrstama pitanja, gdje su pitanja i odgovori unaprijed pripremljeni i fiksni,
- na području dinamičkih pitanja, oba sustava nude elementarne mehanizme za postizanje dinamičnosti uporabom jednostavnih predložnih jezika (engl. templates), što je tipično dostatno za postavljanje pitanja poput "Koliko iznosi {a} uvećano za {b}", pri čemu se točan odgovor definira formulom "{a}+{b}"; primjeri složeniji od ovoga nisu podržani (primjerice, funkcija za izračun Fibonaccijevog broja),
- nema potpore za dinamičku izradu i korištenje višemedijskih sadržaja (dinamički generirane slike, zvukovi i sl.),
- nema potpore za obradu složenih oblika tekstovnih odgovora (koji su strojno obradivi) – primjerice, analiza unesene Booleove funkcije, što se može postići već i primitivnim jezičnim procesorom,
- nema mogućnosti uporabe dijeljenih računalnih resursa za potrebe stvaranja složenih pitanja i provjere točnosti odgovora, pri čemu se pod pojmom složeno pitanje ovdje smatra pitanje za čije je stvaranje i ocjenjivanje potrebno utrošiti znatne računalne resurse i koristiti složene algoritme,
- potpora za definiranje politike kolegija je nedostatna: očekuje se da će se kolegiji prilagoditi mogućnostima ovih alata, a ne obratno; primjerice, nije moguće ostvariti uvjet "test X ćeš rješavati tako dugo dok ga ne položiš, a to ti je preduvjet za pristup testu Y".

Najvažniji dojam koji se dobije kroz ovu analizu jest da se e-ispitivanje u sustavima e-učenja želi riješiti tako da bude za svakoga, što znači da se sve zadatke može unijeti kroz nekoliko klikova mišem i unosom teksta (bez ikakvog učenja i ulaganja dodatnog truda). Međutim, rezultat takvog pristupa su kvizovi s pitanjima koja imaju elementarne mogućnosti, i ništa više, jer je za sve složenije potrebno uložiti veliku količinu vremena, truda i volje.

3.4 Sustav WODLS

Sustav WODLS [25] razvijen je na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu. Sustav je implementiran u okviru aplikacijskog poslužitelja (engl. application server) Lotus DOMINO [26], što automatski povlači i zasnovanost na polustrukturiranoj bazi podataka temeljenoj na dokumentima (engl. semistructured

document based database), a ne na relacijskoj bazi podataka (engl. relational database). Sustav pri tome obilno koristi mogućnosti koje nudi platforma DOMINO, poput čak tri podržana jezika za pisanje agenata (Java, LotusScript te Formula language), agente i sl, što nažalost cijelu aplikaciju čini izuzetno sporom, što je jedan od glavnih razloga zašto je naknadno razvijen sustav StudTest.

Sustav WODLS zamišljen je kao sveobuhvatno rješenje za e-učenje, no nakon otprilike dvije godine rada razvoj je prekinut. Sa stanovišta potpore e-ispitivanju, sustav nudi tri tipa zadataka:

- ABC pitalica s jednim točnim odgovorom, što predstavlja zadatak s *n* ponuđenih odgovora i jednim točnim,
- ABC pitalica s više točnih odgovora, što predstavlja zadatak s *n* ponuđenih odgovora i više točnih, te
- unos broja, što predstavlja zadatak s brojčanim rješenjem.

Zadaci se interno pohranjuju kao XML dokumenti. Dobra je strana ovog sustava mogućnost direktne ugradnje složene logike za stvaranje pitanja kroz definiranje zadataka u programskom jeziku Java. Naime, prva dva tipa zadatka inkapsulirana su kroz sučelja (Slika 3-1 (a) prikazuje sučelje ABC pitalice s jednim točnim odgovorom, a Slika 3-1 (b) sučelje ABC pitalice s više točnih odgovora), čime je omogućena izrada Java razreda koji implementiraju ta sučelja i s kojima sustav može

```
public interface ABCZadatak {
     String napraviNoviZadatak();
     void init(String initData);
     void postaviParametar(String paramIme,
                           String paramVrijednost);
     int dajBrojOpcija();
     int dajTocnuOpciju();
     String dajTekstZadatka();
     Vector dajOpcije();
     void pocisti();
}
```

Slika 3-1. a) sučelje ABC pitalice s jednim točnim odgovorom

```
public interface ABCMZadatak {
     String napraviNoviZadatak();
     void init(String initData);
     void postaviParametar(String paramIme,
                           String paramVrijednost);
     int dajBrojOpcija();
     Vector dajTocnostOpcija();
     String dajTekstZadatka();
     Vector dajOpcije();
     void pocisti();
}
```

Slika 3-1. b) sučelje ABC pitalice s više točnih odgovora

raditi kao s klasičnim zadacima.

Međutim, ovaj je pristup iskazao i određene probleme, zbog kojih je sustav napušten. Naime, u trenutku stvaranja novog pitanja sustav očekuje stvaranje kako pitanja tako i točnog odgovora. Kod ABC pitalica ovakav je pristup prihvatljiv, kao i kod bilo kojeg drugog tipa pitanja gdje student odabire rješenje. Međutim, kod ostalih vrsta pitanja gdje student upisuje odgovor, ovo više nije prihvatljivo; naime, postoje zadaci kod kojih točnih odgovora može biti beskonačno te se ne može očekivati od studenta da upiše baš onaj koji je pohranjen u sustavu.

Sustav WODLS ima jako važnu ulogu u ovom radu, jer je analiza njegovih nedostataka dovela do ideje o razradi specijaliziranog podsustava e-ispitivanja opisanog u ovom radu.

3.5 Sustav CMS Nescume

Na kraju ovog poglavlja još ukratko se još opisuje sustav CMS Nescume (engl. Course Management System) [24] koji se još uvijek razvija za potrebe vođenja kolegija. Sustav ne nudi mogućnost direktnog e-ispitivanja, već se za tu funkciju oslanja na vanjske sustave, u konkretnom slučaju na sustav StudTest koji je prototipna implementacija modela opisanog u ovom radu. Također, sustav ne nudi mogućnost obrazovanja korisnika, što je danas centralni pojam za sustave poput WebCT i MOODLE. Umjesto toga, namjena ovog sustava može se ocijeniti komplementarnom sustavima e-poučavanja (primarno distribucija materijala za učenje u raznim oblicima) i sustavima e-ispitivanja (provjera usvojenog znanja), jer je njegova funkcija omeđena na potporu protoku poslova kolegija (engl. workflow), odnosno e-vođenje kolegija.

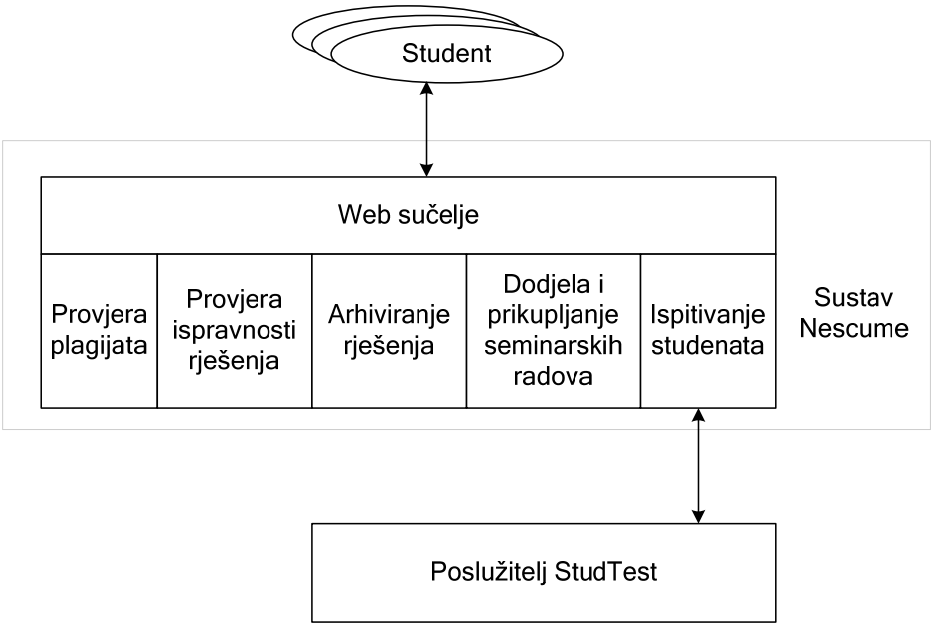
Sustav Nescume definira sljedeće pojmove:

- *kolegij*: apstraktan pojam koji obuhvaća sve semestre ponude tog kolegija kroz niz akademskih godina,
- *primjerak kolegija*: jedna konkretna ponuda kolegija, jednoznačno definirana akademskom godinom i semestrom,
- *korisnik*: svaki student, asistent, nastavnik ili bilo koja druga osoba koja ima neku ulogu u sustavu,
- *grupa*: hijerarhijska zajednica korisnika koja se može sastojati i od drugih podgrupa; primjerice, svaki primjerak kolegija ima primarnu grupu svojih korisnika,
- *aktivnost*: apstraktni pojam koji se odnosi na bilo kakvu aktivnost u okviru kolegija (primjeri aktivnosti su seminarski radovi, domaće zadaće, laboratorijske vježbe); aktivnosti se također mogu hijerarhijski povezivati,
- *laboratorijska vježba*: konkretna laboratorijska vježba koja pripada aktivnosti laboratorij,
- *zadatak laboratorijske vježbe*: apstraktni pojam koji opisuje manju grupu poslova koje treba napraviti kako bi se riješio zadatak; laboratorijska vježba sastoji se od jednog ili više zadataka, pri čemu student može dobiti sve

zadatke ili samo neke, prema određenom kriteriju – primjerice zadnjoj znamenci matičnog broja,

- *posao unutar zadatka laboratorijske vježbe*: ovo predstavlja elementarnu jedinicu posla koju student mora napraviti u okviru nekog zadatka, i odgovara jednoj datoteci koju je student napisao, primjerice, izvorni kod u programskom jeziku C, opis digitalnog sklopa u jeziku VHDL, datoteka dokumentacije napisana u alatu Microsoft Word ili Open Office; zadatak može definirati više poslova koje treba obaviti; zadatak je riješen samo ako su svi poslovi riješeni; posao je riješen kada se odgovarajuća studentova datoteka ukrca (engl. upload) i zaključa na poslužitelj,
- *test*: provjera znanja koja u sustavu može biti vezana uz neku aktivnost, primjerice, domaće zadaće, gdje je svaka zadaća zapravo jedan test ili uz neku laboratorijsku vježbu, što omogućava definiranje raznih vrsta testova na ulazu u laboratorij te prije izlaska iz laboratorija.

Sustav Nescume korišten je za laboratorijske vježbe i domaće zadaće na kolegiju Digitalna logika izvedenom u akademskoj godini 2005/2006 u zimskom semestru. Na ponovljenom izdanju kolegija u ljetnom semestru iste godine sustav je korišten još i za poslovanje seminarskim radovima studenata.



Slika 3-2. Odnos sustava Nescume i StudTest

Sustav podržava tri važna proširenja vezana uz laboratorijske vježbe:

- *mogućnost detekcije plagijata*, čime se želi spriječiti studente da međusobno prepisuju rješenja laboratorijskih vježbi te ih prisiliti na samostalan rad, te
- *automatska provjera studentskih rješenja*, čime se bitno može olakšati proces vrednovanja rada studenta na vježbi, a i samom studentu se može pružiti direktna povratna informacija o kvaliteti njegovog rješenja i to od elementarne informacije tipa "radi-ne radi" do ispisa analizatora koda poput

splint-a za programski jezik C, koji će pomno pregledati kod u potrazi za uporabom nesigurnih funkcija i sl. i to dokumentirati na svom izlazu,

- *arhiviranje svih rješenja na jednom mjestu*, čime se može izgraditi baza rješenja za dotični kolegij kroz više generacija studenata.

Trenutno je u tijeku još nekoliko velikih proširenja sustava koja će omogućiti još veću fleksibilnost i bolje preklapanje mogućnosti sustava sa stvarnim zahtjevima kolegija. Dio planiranih proširenja obuhvaća razvoj podsustava za razmjenu poruka između asistenata i studenata, podsustav za poslovanje prijavama i žalbama, podsustav za poslovanje bodovima studenata (obzirom na niz izvora bodova), podsustav za poslovanje grupama studenata temeljen na principu burze, gdje bi studenti samostalno mogli razmjenjivati grupe. Osim toga, velika je pažnja posvećena i ugodnom radu u sustavu te smanjenju stresa koji uporaba ovakvog sustava može uzrokovati kod studenata. Stoga je dolična pažnja posvećena i detaljima poput omogućavanja korištenja njima poznatih zaporki i korisničkih imena s drugih fakultetskih sustava za pristup sustavu Nescume, pri čemu se ovjera u pozadini obavlja kroz te sustave.

Sustav je implementiran uporabom otvorenih tehnologija kao što su programski jezik Java te baza podataka MySql za pohranu informacija. Slika 3-2 prikazuje odnos sustava Nescume i prototipne implementacije sustava StudTest-a.

Tablica 3-1 prikazuje usporedbu značajnih karakteristika sa stanovišta e-ispitivanja prethodno prikazanih sustava. Iz tablice je jasno vidljiva prednost sustava zasnovanog na modelu StudTest.

Tablica 3-1. Usporedba sustava WebCT, MOODLE, WODLS i StudTest

	WebCT	MOODLE	WODLS	StudTest
Statička pitanja raznih tipova	DA	DA	DA	DA[1]
Uporaba dinamičkih višemedijskih sadržaja	NE	NE	NE	DA
Jednostavni dinamički zadaci	DA	DA	NE	DA
Složeni dinamički zadaci	NE	NE	NE	DA
Složeni algoritmi bodovanja provjere znanja	NE	NE	NE	DA
Potpora definiranju politike složenijeg sveučilišnog kolegija	NE	NE	NE	DA

[1] Broj podržanih tipova nije ograničen, i lako je dodavanje novih tipova. Kod ostalih sustava gdje izvorni kod nije dostupan, ili gdje lagana proširivost nije uzeta u obzir, ovo nije moguće izvesti, ili jest ali uz veliki utrošak resursa.

4 Model e-ispitivanja StudTest

U ovom se poglavlju definira model e-ispitivanja StudTest. No prije same definicije modela razmatraju se zahtjevi koji se postavljaju na sustave e-ispitivanje te posebice ideja *višestruke iskoristivosti zadataka*. Nakon toga, definiraju se osnovni koncepti modela, konstante i ostali pomoćni objekti modela. Na kraju poglavlja opisuje se potpora modela za dinamičko generiranje višemedijskih sadržaja u funkciji obogaćivanja samih zadataka.

4.1 Zahtjevi na sustav e-ispitivanja

Prije no što je započeo razvoj sustava StudTest (kako nacrta modela, tako i početne prototipne implementacije), obavljena je analiza u svrhu spoznaje kakvo je trenutno stanje i kakve su potrebe na kolegijima fakultetske razine u smislu računalom podržane provjere znanja studenata. Za potrebe elektroničke provjere znanja na raspolaganju je bio sustav WODLS [25], koji je dvije akademske godine korišten za provjeru znanja studenata u okviru laboratorijskih vježbi kolegija Digitalna elektronika. Kroz implementaciju i rad s ovim sustavom stečena su vrlo važna iskustva, kako u pogledu samih tehnologija, tako i kroz utjecaj istih na studente.

Analiziran je određen broj kolegija (Digitalna elektronika, Digitalna logika, Mreže računala, Inteligentni sustavi, Neizrazito, evolucijsko i neuroračunarstvo, Strojno učenje, Arhitektura i organizacija računala te Formalni postupci u oblikovanju računalnih sustava) pri čemu je želja bila vidjeti kakva su pitanja prikladna za te kolegije, odnosno što je sve potrebno podržati.

Zaključci su sljedeći. Vezano uz mehanizme za stvaranje i rad sa zadacima, potrebno je osigurati sljedeće:

- *potrebno je podržati statička pitanja*; statička su pitanja ona kod kojih se tekst pitanja i odgovori definiraju unaprijed; kao ilustracija ove vrste pitanja može poslužiti razredbeni ispit: unaprijed se priprema određen broj grupa testova, i unutar grupe svi studenti dobivaju identična pitanja,
- *potrebno je podržati dinamička pitanja*; dinamička su pitanja ona kod kojih se dijelovi teksta pitanja, a možda i odgovora, generiraju posredstvom slučajnog mehanizma; u ovom kontekstu na dinamičko se pitanje može gledati kao na masku (engl. template) koju prije uporabe treba do kraja specijalizirati; ovime svaki student može dobiti zadatak iste vrste, ali opet različit,
- *potrebno je podržati zadatke koji nemaju jedinstveno točno rješenje*; ovisno o gradivu koje se želi ispitati, često se mogu koristiti zadaci koji će provjeriti razumije li student određene koncepte, tako da sam unosi rješenje (primjerice, u obliku teksta); moguće su situacije u kojima student može upisati jedno od konačno ili čak beskonačno mnogo točnih rješenja (primjer prvoga je "Koji broj potenciran na drugu potenciju rezultira brojem 4?", a drugi "Na satu ste učili razliku između parnih i neparnih brojeva. U prostor za unos rješenja unesite jedan paran broj."); iz ovoga slijedi da protok poslova vezanih uz upravljanje zadacima mora imati barem dvije faze: fazu stvaranja zadatka te fazu ocjenjivanja zadatka,

- *zadaci moraju podržavati uporabu višemedijskih sadržaja*; često je potrebno ispitati znanje o nečemu prikazanom primjerice slikom: "Koju funkciju obavlja digitalni sklop prikazan slikom?", ili "BDD[2] na slici prikazuje neku funkciju. O kojoj se funkciji radi?".
- *zadaci moraju podržati uporabu dinamičkog višemedijskog sadržaja*; vrlo često je potrebno osigurati mogućnost generiranja višemedijskog sadržaja koji će se razlikovati za svakog studenta, a bit će definiran u trenutku stvaranja samog zadatka.
- *potrebno je ponuditi mehanizam kojim će više zadataka moći koristiti usluge specijaliziranih alata*; primjer toga je programska podrška koja na temelju opisa sklopa (primjerice strukturni model digitalnog sustava opisan u jeziku VHDL) može generirati sliku sa shemom sklopa. Takav se alat može koristiti u nizu pitanja vezanih uz brojila, registre, strojeve s konačnim brojem stanja i sl.
- *potrebno je osigurati mehanizam temeljem kojeg će se obavljati unos složenih rješenja zadataka* (primjerice, kod zadatka koji zahtijeva da se nacrta shema sklopa koji obavlja zadanu funkciju),
- *potrebno je razdvojiti rješavanje načina interakcije studenta i samog zadatka od logike zadatka*, tako da se stvaranje novih zadataka svodi na pisanje samo onih dijelova koda koji su doista specifični za taj zadatak. Na ovaj se način želi osigurati nezavisnost prikaza zadatka od samog zadatka.

Vezano uz pristup samim provjerama znanja potrebno je osigurati sljedeće:

- *rješavanje provjere odjednom*; karakteristika ovog načina jest da student pristupa provjeri znanja, rješava zadatke u skladu s pravilima provjere te nakon određenog vremena završava rješavanje; provjera se ocjenjuje i student dobiva povratnu informaciju u obliku bodova, uvid u samu zadaću, točna rješenja (gdje je odgovoreno pogrešno) te objašnjenja (zašto je njegov odgovor pogrešan),
- *rješavanje provjere s prekidima, kroz duži vremenski interval*; karakteristika ovog načina jest da student može pristupiti provjeri znanja, prepisati si dio zadataka, zamrznuti provjeru (engl. suspend), rješavati zadatke na papiru, nastaviti provjeru, unijeti rješenja, prepisati nove zadatke itd.; u ovom slučaju zadaća se na ocjenjivanje šalje kada je student samostalno pošalje ili kada nastupi neko od ograničenja (primjerice, ako je zadano vremensko ograničenje),
- *osigurati odgovarajuće sigurnosne mehanizme*; potrebno je osigurati da se provjeri može pristupiti samo u slučaju u kojem vrijede sva prethodno definirana sigurnosna ograničenja; primjeri ovih ograničenja mogu biti: pristup uz poznavanje zaporke, pristup samo s određenih IP adresa i sl.,
- *potrebno je osigurati potporu za provođenje politike kolegija*; vezano uz provjere znanja, kolegiji mogu definirati različite politike, poput vremenskih ograničenja (npr. ulazni se test piše 10 minuta a izlazni 12 minuta), kriterija

[2] BDD (engl. Binary Decision Diagram), dijagram binarne odluke

za prolaz (ulazni test smatrat će se položenim ako se osvoji barem 40% bodova – oni koji to ne osvoje moraju ulazni test pisati ponovno, sve dok ne zadovolje prethodni uvjet), preduvjeta (ne može se pristupiti izlaznom testu ako ulazni test nije položen) i sl.

Vezano uz opće karakteristike samog sustava, potrebno je osigurati:

- *podesivost* (engl. scalability) što tipično znači da sustav mora podržavati mehanizme raspodijeljenosti, kako bi se problem zagušenosti mogao riješiti dodavanjem novih poslužitelja koji prema korisniku izgledaju kao jedan uniformni sustav,
- *dobar odziv na kratkotrajna visoka vršna opterećenja*; sustavi e-ispitivanja imaju jedan karakterističan uzorak raspodjele opterećenja koji je rezultat načina uporabe; kada se koristi u okviru provjera znanja pod nadzorom zaduženog osoblja, tada svi studenti tipično istovremeno zahtijevaju provjeru (na znak "sad!"), odnosno istovremeno predaju provjeru na ocjenjivanje (primjerice, po isteku desete minute); ovo je vrlo važna osobina koja direktno utječe na percepciju sustava kod studenata[3],
- *podršku za različite tehnologije pristupa sustavu*; danas se sustavima e-učenja uobičajeno pristupa putem Weba; međutim, popularizacijom mobilnih tehnologija javlja se i paradigma m-učenja za koju također treba osigurati odgovarajući podršku u sustavima e-ispitivanja, a koja više ne mora biti u obliku standardnih HTML stranica (jer takvi uređaji mogu, primjerice, imati zaslone lošijih mogućnosti, manjih dimenzija i sl.).

Imajući u vidu ove zahtjeve, kao i iskustvo stečeno kroz rad sa sustavom WODLS, u nastavku se definira model sustava StudTest, koji na odgovarajući način uvažava sva prethodna razmatranja.

4.2 Ideja višestruke iskoristivosti zadataka

Temeljna ideja na kojoj je sagrađen model StudTest jest izgraditi zadatke po uzoru na servlete [27], popularno rješenje problema dinamičkog generiranja Web sadržaja. Servleti su mali Java programi koji "žive" na Web poslužitelju, u njegovom posebnom dijelu koji se zove sadržatelj servleta (engl. servlet container). Sadržatelj servleta upravlja životnim ciklusom servleta, i koristi servlet za dinamičko generiranje HTML stranica. Problem upravljanja zadacima riješit će se na analogan način: zadatak je potrebno promatrati kao atomarnu komponentu koja ima svoje sučelje i životni ciklus te njime upravlja *sadržatelj zadataka*.

Obzirom na činjenicu da je u kontekstu ovog rada zadatak samostalna atomarna jedinica, koja studenta ispituje točno jedan pojam, svaki zadatak (koji se može sastojati od više datoteka) pakira se u zasebnu arhivu, dobiva jedinstven identifikator i opis arhive (*manifest*). Na ovaj se način zadaci mogu čuvati u odgovarajućem repozitoriju zadataka, mogu se dohvaćati s bilo koje lokacije i ugraditi u

[3] Iskustvo stečeno sa sustavom WODLS jasno pokazuje što se događa kada sustav ima loš odziv na ovu vrstu opterećenja, što će biti ilustrirano na primjeru. U sustavu je bilo definirano da se zadaća rješava 10 minuta (i toleriralo se je kašnjenje od 1 minute). U desetoj minuti svi studenti su istovremeno poslali zadaće na ocjenjivanje – i sustav se je zagušio. Zadaće koje je uspio ocijeniti unutar minute je i prihvatio; ostale je poništio, iako su bile poslane na vrijeme.

odgovarajući sadržatelj zadataka. Na sličan je način u normi SCORM po analogiji definiran SCO (komponenta koja predstavlja elementarnu granulu znanja, a s poslužiteljem komunicira putem sučelja definiranog jezikom JavaScript).

Ovdje definirani model vrlo malo odstupa od opisane ideje: zadatak neće biti potpuno samostalna jedinica, jer je jedan od zahtjeva utvrđenih u točki 4.1 da zadatak mora moći komunicirati s alatima koji nude specijalizirane usluge. Ovi alati neće biti sastavni dio samog zadatka, što povlači potrebu za pohranom ovisnosti o takvim alatima u posebnu datoteku (manifest). Ovo također znači da će prilikom uvoza takvog zadatka sadržatelj zadataka prvo provjeriti ima li on na raspolaganju sve potrebne specijalističke alate, a tek u slučaju potvrdnog odgovora krenuti u pohranjivanje zadatka. Definirani model tako uvodi sloj apstrakcije nad alatima, koji će tada postati zamjenjivi. Svi takvi specijalistički alati tada postaju dostupni iz raznih repozitorija pa se i oni mogu dinamički dodati u sustav.

4.3 Pisanje provjere znanja

Prije opisa modela razvijenog sustava, pogledajmo kako izgleda proces provjere znanja izveden klasičnim putem, odnosno kako bi (hipotetski) trebao izgledati proces provjere znanja na računalu.

Klasična provjera znanja sastoji se od tri koraka. U prvom koraku sastavlja se provjera znanja, što obuhvaća odabir i pripremu zadataka te tiskanje i/ili kopiranje potrebnog broja primjeraka provjere. Najčešće svi studenti rješavaju istu provjeru; rjeđe, kako bi se spriječilo prepisivanje, izrađuje se nekoliko različitih grupa zadataka. Također, potrebno je obaviti rezervaciju dvorana i ljudi (nadzornika) za čuvanje studenata te sastaviti popise studenata koji smiju pristupiti provjeri, u skladu s politikom kolegija. Do trenutka pisanja provjere znanja same se provjere čuvaju na sigurnom mjestu.

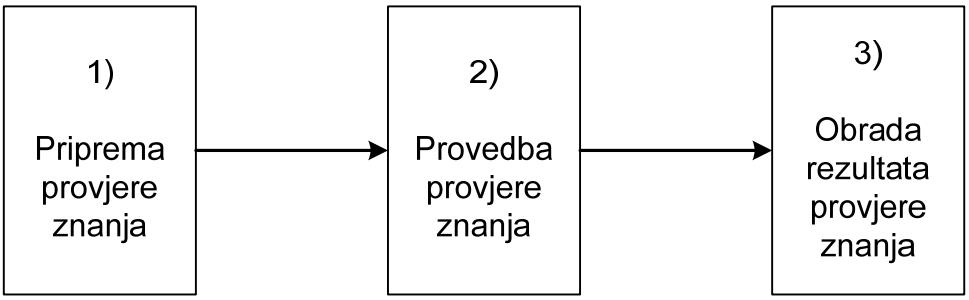
Provedba postupka provjere znanja predstavlja drugi korak. Ovo obuhvaća sigurno dostavljanje provjera u dvoranu, podjelu provjera studentima te nadzor pisanja provjere znanja, kako bi se spriječilo korištenje nedopuštenih sredstava i aktivnosti. Po završetku termina za provjeru znanja provjere se prikupljaju, i sigurno pohranjuju.

Treći korak predstavlja proces ispravljanja i vrednovanja provjere znanja, bodovanja same provjere, provedba postupka uvida u provjere i eventualne žalbe. Slika 4-1 prikazuje korake klasične provjere znanja.

U prvom koraku tipično sudjeluje manji broj ljudi (nastavnici i tipično jedan asistent). Drugi korak predstavlja najzahtjevniji dio procesa, kako sa stanovišta ljudskih, tako i prostornih resursa. Naime, za nadzor je potrebno angažirati velik broj nadzornika, a budući da se želi spriječiti prepisivanje, studenti se u dvorane razmještaju vrlo rijetko, što znači da je potrebno zauzeti velik broj dvorana. Treći korak je tipično najdugotrajniji, jer je potrebno pregledati sve provjere znanja te svaki zadatak svake provjere objektivno ocijeniti.

Ukoliko se za potrebe provjere znanja iskoristi računalo, dolazi se do sljedećeg hipotetskog scenarija.

U prvom koraku sastavlja se provjera znanja, što obuhvaća odabir i pripremu zadataka. Zadaci se biraju iz pripremljenog repozitorija zadataka. Neki zadaci imaju mogućnost automatskog vrednovanja, dok neki nemaju. S druge strane, dio zadataka

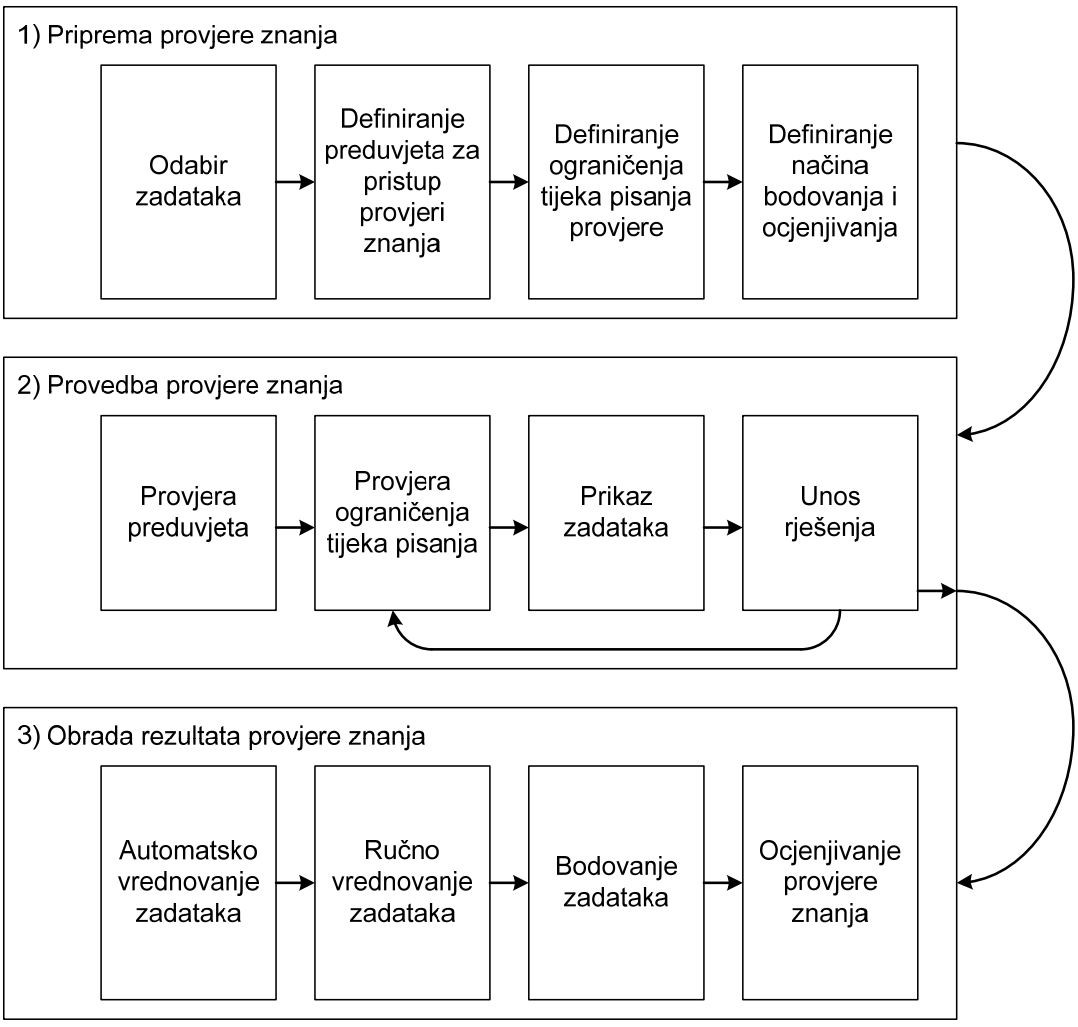


Slika 4-1. Klasična provjera znanja

je statički, dok je dio dinamički. Odabire se točan broj zadataka koji će ući u provjeru, ili se odabire veći skup zadataka od kojih će svaki student dobiti neki podskup. U sklopu ovog koraka odabiru se ograničenja koja je potrebno provesti (primjerice, vremenska ograničenja na duljinu pisanja provjere znanja), način bodovanja zadataka (svi zadaci nose isti broj bodova ili ne), uvjeti za pristup provjeri znanja i sl. Također, potrebno je obaviti rezervaciju dvorana i ljudi (nadzornika) za čuvanje studenata. Do trenutka pisanja provjere znanja same se provjere čuvaju u elektroničkom obliku na sigurnom mjestu (na poslužitelju).

Drugi korak predstavlja provedba postupka provjere znanja. Radi nadzora, studenti se smještaju u dvorane opremljene računalima, gdje su računala spojena na Internet. Nakon prijave studenata na sustav za provođenje provjere znanja, za svakog studenta sustav odabire potreban broj zadataka, i stvara konkretna pitanja (za dinamičke zadatke). Potom nadzornik obznanjuje zaporku za pristup provjeri znanja, i studenti od sustava zahtijevaju pristup provjeri znanja. Sustav provjerava ispunjava li student definirana ograničenja, odnosno smije li pristupiti provjeri. Budući da je u opisniku provjere znanja definirano da se provjeri pristupa samo uz poznavanje zaporke, i sa zadanih IP adresa, sustav studentima privremeno uskraćuje pristup provjeri, i pita ih za zaporku. Nakon što student upiše zaporku, sustav ponavlja postupak provjere, i utvrđuje da student zadovoljava sva definirana ograničenja (uključivo pristup sa zadanih IP adresa) te smije pristupiti provjeri. Sustav provjerava ima li kakvih ograničenja na tijek pisanja provjere znanja, i pronalazi vremensko ograničenje od 30 minuta. Zbog toga se studentu prikazuju prva dva zadataka (jer je u opisniku provjere definirano da se zadaci prikazuju dva po dva), i sat koji odbrojava od 10 minuta prema nuli. Iz istog razloga, studentu se ne prikazuje tipka "Povratak na prošle zadatke". Student upisuje odgovore na ponuđene zadatke. Obzirom da je tako definirano u opisniku provjere, sustav je uz svaki zadatak stvorio element korisničkog sučelja kojim student može odabrati koliko je siguran u svoje rješenje. Za prvi zadatak odabire 1 (apsolutno siguran) dok za drugi zadatak odabire 0,7. Odabirom tipke "Dalje" student od sustava zahtijeva sljedeća dva zadatka. U trenutku kada sustav primi taj zahtjev, najprije se provjerava ima li student još uvijek pravo na rješavanje provjere, tj. jesu li još uvijek zadovoljena sva ograničenja postavljenja na tijek pisanja provjere znanja (ili je primjerice isteklo dozvoljeno vrijeme). Kako je sve zadovoljeno, sustav odabire sljedeća dva zadatka, i prikazuje ih studentu (pri čemu ažurira sat koji prikazuje preostalo vrijeme). Student unosi rješenja i mjeru pouzdanosti te zahtjeva sljedeće zadatke (čime se postupak ponavlja). S 9. i 10. zadatkom student ima problema, i ne zna kako ih riješiti. U jednom trenutku ističe vrijeme, i klijent kroz koji student pristupa provjeri znanja automatski šalje provjeru

sustavu (simulira pritisak na tipku "Gotovo"). Sustav zaprima zahtjev za završetkom pisanja provjere, i obzirom da je on stigao unutar zadanog vremenskog razdoblja, prihvaća zahtjev. Kod svih zadataka koji podržavaju automatsko vrednovanje, pokreću se vrednovatelji. Tri zadatka automatsko vrednovanje ne podržavaju, te su poslana asistentu. Drugi je student po dolasku na zadnji zadatak onemogućio svom klijentu kontaktiranje poslužitelja, kako bi imao više vremena za rješavanje provjere. U trenutku kada je riješio zadatke, omogućio je klijentu pristup poslužitelju, te šalje svoju provjeru. Na poslužitelju se najprije pristupa provjeri ograničenja tijeka pisanja provjere znanja te se utvrđuje primitak zahtjeva za završavanjem provjere nakon što je prošlo vremensko ograničenje, pa se zbog toga zadaća poništava.



Slika 4-2. Provjera znanja u sustavu e-ispitivanja

Treći korak predstavlja proces ispravljanja i vrednovanja provjere znanja, bodovanja same provjere, provedba postupka uvida u provjere i eventualne žalbe. Dio vrednovanja zadataka već je automatski obavljen po završetku drugog koraka. Kako bi ovaj postupak proveo do kraja, asistent se povezuje na sustav koji mu nudi na vrednovanje zadatak po zadatak. Nakon što asistent obavi vrednovanje svih zadataka jedne provjere koje trebalo ručno vrednovati, sustav pokreće proces bodovanja same provjere i izračuna ukupnog broja bodova. Kako je u opisniku definirano da se bodovanje obavlja proporcionalno mjeri točnosti zadatka i mjeri pouzdanosti

zadatka, prvi student iz primjera, koji je riješio točno pola prvog zadatka i 75% drugog zadatka, za prvi zadatak dobiva 5 bodova (10*0,5*1,0=5), a za drugi zadatak 5,25 bodova (10*0,75*0,7=5,25, svaki zadatak nosi maksimalno 10 bodova).

Od trećeg koraka klasične provjere znanja sustav ne može automatski rješavati studentske žalbe. Međutim, ukoliko su zadaci ispravno napravljeni, tada po definiciji ne bi smjelo biti žalbi.

Slika 4-2 prikazuje korake provjere znanja u sustavu e-ispitivanja.

4.4 Koncepti modela StudTest

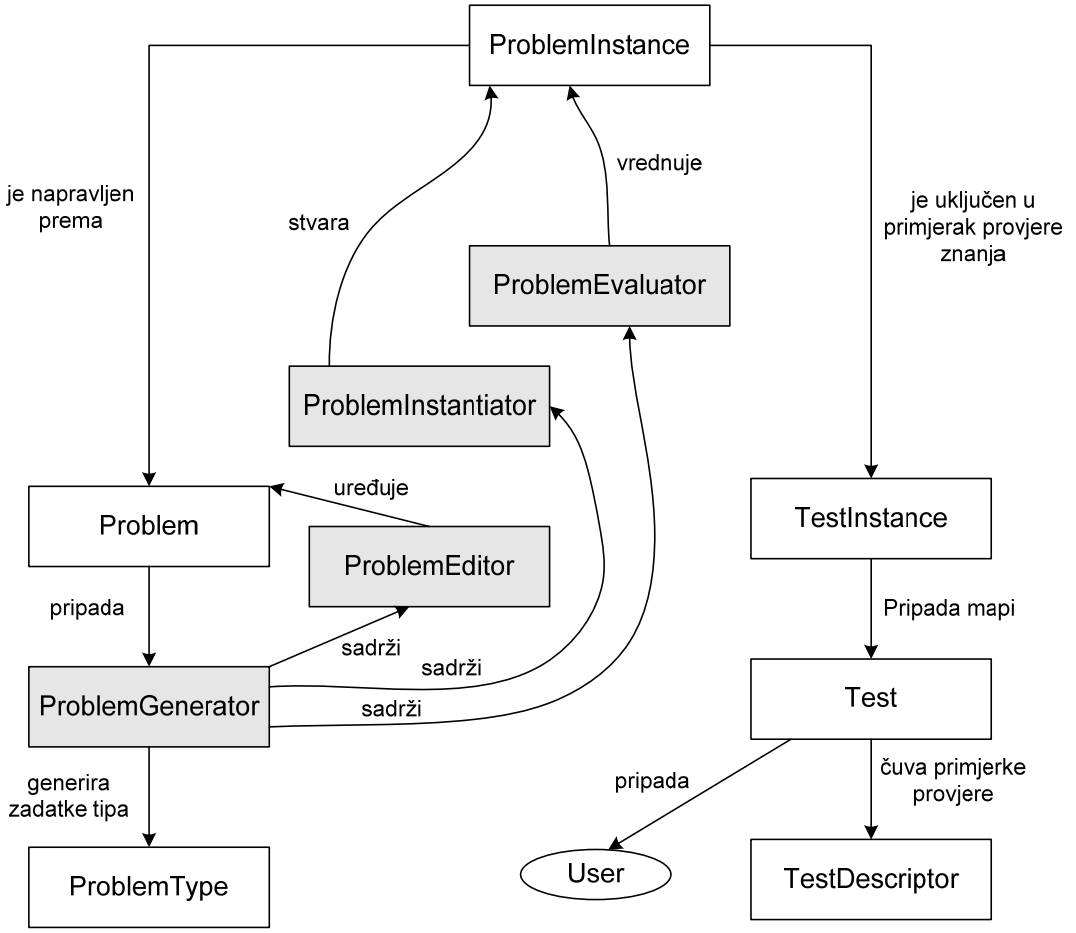
Osnovni pojam koji model definira jest komponenta koja konceptualno predstavlja zadatak, i koja se u analogiji sa servletima naziva *prlet*. Uloga prleta je poslovanje zadacima, što obuhvaća stvaranje i podešavanje šablone po kojoj će se stvarati zadaci za korisnike, stvaranje samih zadataka na temelju šablona, vrednovanje korisnikovih odgovora, generiranje obrazloženja netočnosti odgovora te generiranje točnog odgovora na postavljeno pitanje, kako bi korisnik provjeru znanja naknadno mogao koristiti i za učenje. Model StudTest za pružanje usluga prletima nudi i niz potpornih koncepata koji će prletima omogućiti dijeljenje složenih algoritama te dinamičku izradu višemedijskih sadržaja. StudTest se tada definira kao *sadržatelj prleta* (tj. prlet container). Prlet ima globalno jedinstveni identifikator, čime postaje globalno dostupan, pa se otvara mogućnost uvoza u druge sustave, kao i svu potrebnu logiku za uređivanje parametara prleta, stvaranje konkretnog pitanja koje će biti postavljeno studentu te opcionalno logiku za ocjenjivanje studentovog odgovora. Ovo je posljednje opcionalno, jer će kod nekih standardnih tipova zadataka funkcionalnost obavljati ocjenjivači ugrađeni u sam sustav.

Pod pojmom osnovni koncepti podrazumijevaju se oni koncepti sustava koji služe za modeliranje provjere znanje i prikaza samih zadataka. Model definira sljedeće koncepte (u zagradama se nalaze izvorni nazivi koncepata, koji će prilikom modeliranja u objektno orijentiranoj paradigmi odgovarati imenima razreda):

- opisnik provjere znanja (TestDescriptor),
- test (Test),
- primjerak testa (TestInstance),
- tip zadatka (ProblemType),
- prikazivač zadatka (ProblemRenderer),
- prikazivač provjere znanja (TestRenderer),
- generator zadatka (ProblemGenerator),
- uređivač zadatka (ProblemEditor),
- stvaratelj primjerka zadatka (ProblemInstantiator)
- vrednovatelj zadatka (ProblemEvaluator).

Slika 4-3 prikazuje međusobni odnos dijela koncepata usko vezanih uz prlet (zasivljene kućice) te provjeru znanja. Veze između koncepata su usmjerene, i treba

ih čitati u naznačenom smjeru. Primjerice: "ProblemInstance je napravljen prema Problem".



Slika 4-3. Međusobni odnos dijela koncepata modela StudTest

TestDescriptor konceptualno predstavlja jednu provjeru znanja (primjerice, 1. kontrolna zadaća iz kolegija Digitalna logika, akademska godina 2005/2006). Kroz TestDescriptor se obavlja opće opisivanje same provjere znanja – primjerice, koliko će imati pitanja, hoće li imati vremenska ograničenja, koji sigurnosni mehanizmi će biti aktivirani tijekom pisanje provjere, koliko će nositi bodova, koja će biti politika ocjenjivanja (samo pozitivni bodovi, pozitivni-negativni bodovi, proporcionalno bodovanje kod kojeg je broj bodova proporcionalan mjeri točnosti rješenja, bodovanje uz pouzdanost kod kojeg je broj bodova proporcionalan mjeri pouzdanosti studenta, itd.) i sl. Tablica 4-1 prikazuje sve elemente koji pripadaju ovom konceptu.

Test predstavlja košuljicu u kojoj se nalaze svi pokušaji rješavanja provjere znanja nekog korisnika za neki opisnik provjere znanja. To znači da će za svaki opisnik provjere znanja i za svakog korisnika postojati po jedan objekt Test. Tablica 4-2 prikazuje sve elemente koji pripadaju ovom konceptu.

Tablica 4-1. Elementi koncepta TestDescriptor

TestDescriptor		
Tip elementa	Naziv elementa	Opis elementa
IRepositories	repositories	repozitorij u koji se pohranjuju svi podaci specifični za ovaj opisnik provjere znanja
String	title	naslov provjere znanja
String	description	opis provjere znanja
Integer	autoCreationPolicy	zastavica koja određuje kada će se obaviti ponovno stvaranje primjerka testa za ispitanika; definira skup prihvatljivih vrijednosti

Tablica 4-2. Elementi koncepta Test

Test		
Tip elementa	Naziv elementa	Opis elementa
TestDescriptor	testDescriptor	kojem opisniku provjere znanja pripada ova provjera
List<TestInstance>	testInstances	lista svih primjeraka provjera znanja koje su stvorene za određenog ispitanika i ovaj opisnik provjere znanja
User	user	ispitanik

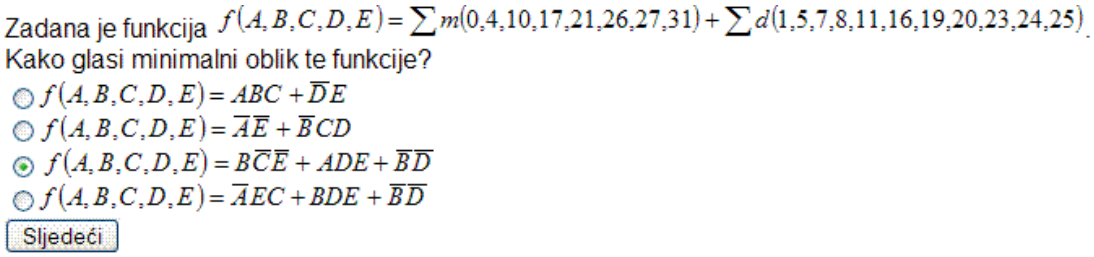
TestInstance je jedan konkretan primjerak provjere znanja koji se isporučuje studentu. Ovisno o politici kolegija, najčešće će za jedan opisnik provjere znanja postojati za svakog korisnika samo po jedan primjerak provjere znanja (te će odgovarajući Test za svakog studenta imati upravo taj jedan primjerak provjere znanja). Razlika između Test i TestInstance može se objasniti uz primjerice sljedeći uvjet: "ulazni test ćeš rješavati sve dok ga jednom ne položiš". Student može ulazni test jednom pasti, pa će mu se stvoriti novi; ako njega padne, ponovno će mu se stvoriti novi; ako iz trećeg pokušaja uspije položiti, to znači da će na kraju njegov Test (koji predstavlja mapu svih njegovih pokušaja) sadržavati tri primjerka provjere znanja(TestInstance). Tablica 4-3 prikazuje sve elemente koji pripadaju ovom konceptu.

ProblemType, odnosno tip zadatka, predstavlja model načina na koji ispitanik doživljava sam zadatak, odnosno metodu njegovog rješavanja. Primjerice, zadatak se može riješiti tako da se odabere jedna od ponuđenih opcija, ili tako da se označe sve točne opcije, ili tako da se rješenje upiše u prostor za unos rješenja, ili pak tako da se nešto nacrta itd. Pri tome je važno napomenuti da tip problema nema nikakve veze s gradivom koje zadatak ispituje. Naime, moguće je sastaviti više zadataka koji

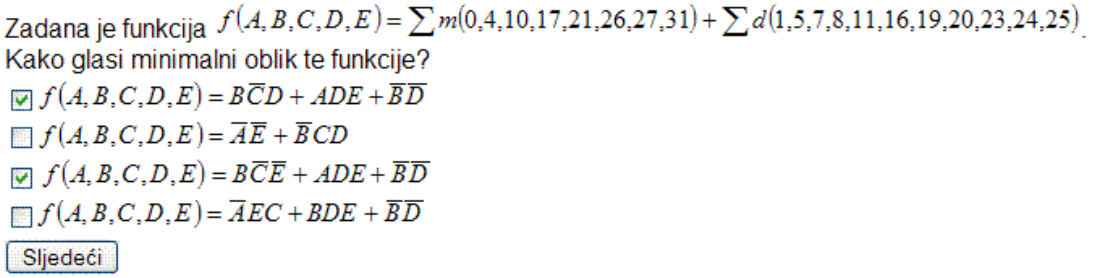
Tablica 4-3. Elementi koncepta TestInstance

TestInstance		
Tip elementa	Naziv elementa	Opis elementa
Test	test	provjera kojoj pripada ovaj primjerak provjere znanja
List<ProblemInstances>	problemInstances	lista svih primjeraka zadataka koji čime ovaj primjerak provjere znanja
IRepositories	repositories	repozitorij za pohranu podataka specifičnih za ovaj primjerak provjere znanja
Integer	status	status provjere znanja. prikazuje skup prihvatljivih vrijednosti
Double	score	ukupni broj bodova dobivenih na ovom primjerku provjere znanja

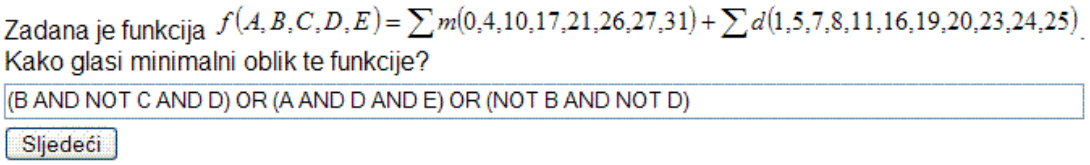
ispituju iste koncepte, ali ispitaniku omogućavaju unos odgovora na različite načine čime se oni razlikuju samo po tipu. Slika 4-4, Slika 4-5 i Slika 4-6 jasno ilustriraju ovu razliku. Sva tri zadatka provjeravaju minimizaciju iste Booleove funkcije. Slika 4-4 prikazuje primjer zadatka gdje se odabire jedan od ponuđenih odgovora. Slika 4-5 nudi više potencijalno točnih odgovora. Slika 4-6 prikazuje tip zadatka u kojem student unosi točan odgovor u obliku teksta.



Slika 4-4. Primjer zadatka s jednim točnim odgovorom



Slika 4-5. Primjer zadatka s više točnih odgovora



Slika 4-6. Primjer zadatka s unosom teksta

Tablica 4-4 prikazuje sve elemente koji pripadaju ovom konceptu.

Tablica 4-4. Elementi koncepta ProblemType

ProblemType		
Tip elementa	Naziv elementa	Opis elementa
String	typePN	javno ime ovog tipa zadatka; mora biti globalno jedinstveno, te se preporuča uporaba URI-ja

ProblemType propisuje 4 metode koje svaki zadatak mora implementirati, što predstavlja standardno sučelje kojim će okolina komunicirati s zadacima. To su redom metode za dohvat pomoći, metoda za dohvat objašnjenja rezultata vrednovanja zadatka (primjerice, ako je nešto pogrešno – zašto je pogrešno), te metode za provjeru podržava li zadatak dohvat točnog rješenja, i ako podržava, je li ono jedinstveno (Tablica 4-5). Odmah se može uočiti da model ne propisuje način dohvata točnog rješenja. Naime, ovisno o tipu zadatka, točno rješenje može biti broj, tekst ili nešto treće. Ono što je propisano su metode koje su neovisne o konkretnom tipu.

Tablica 4-5. Metode koje zadatak mora implementirati kao podršku tipu zadatka

`String getHelp();` Vraća pomoć koju treba ispisati ispitaniku prilikom rješavanja zadatka.
`String getExplanation();` Ukoliko ispitanikovo rješenje nije točno, ova metoda vraća objašnjenje zašto nije točno.
`boolean isCorrectSolutionAvailable();` Metoda provjerava je li generator zadatka u mogućnosti stvoriti primjerak točnog rješenja.
`boolean isCorrectSolutionUnique();` Metoda provjerava je li rješenje koje će generator stvoriti jedinstveno ili nije.

ProblemRenderer odnosno prikazivač zadatka jest komponenta zadužena za stvaranje prikaza određenog tipa zadatka u zadanoj tehnologiji. Primjerice, ispitanik sustavu može pristupati putem Web preglednika, gdje je ABC pitalicu s jednim točnim odgovorom moguće ostvariti uporabom HTML formulara (Slika 4-7 prikazuje primjer formulara za zadatak s unosom teksta), ili pak putem dlanovnika na

kojem HTML možda baš i nije tehnologija izbora, te će pitanje trebati prezentirati na neki drugi način. Ideja iza razdvajanja tipa zadatka i prikazivača zadatka jest postići neovisnost između prleta (tj. logike za upravljanje zadatkom) i prikaza u ciljnoj tehnologiji, što se u prototipnoj implementaciji pokazalo vrlo uspješnim.

```
<form>
  <table width="55%">
    <tr><td>
      Zadana je funkcija <img src="f1.gif" border="0">.
      Kako glasi minimalni oblik te funkcije?
    </td></tr>
    <tr><td><input type="text" size="107"></td></tr>
    <tr><td><input type="submit" value="Sljedeći"></td></tr>
  </table>
</form>
```

Slika 4-7. HTML kod zadatka s unosom teksta

ProblemRenderer propisuje četiri obavezne metode koje moraju biti implementirane (Tablica 4-6).

Tablica 4-6. Metode prikazivača zadatka (ProblemRenderer)

`boolean renderProblem(`
`  ProblemInstance problemInstance,`
`  HashMap properties,`
`  HashMap output,`
`  String purpose);`
Zadatak metode je stvoriti opis predanog primjerka zadatka u tehnologiji koju prikazivač podržava, a u svrhu određenu parametrom purpose (moguće svrhe su: za rješavanje, samo za čitanje, točno rješenje). Dijelovi opisa pohranjuju se u tablicu raspršenog adresiranja (output), a dodatni parametri predaju se kroz tablicu raspršenog adresiranja (properties).

`boolean extractProblemSolution(`
`  ProblemInstance problemInstance,`
`  HashMap properties,`
`  Properties data);`
Svrha ove metode jest upravo suprotno od prethodne – očitati što je ispitanik upisao kao odgovor, i to pohrani u predani primjerak zadatka. Dodatni parametri predani su kroz tablicu raspršenog adresiranja (properties), a ispitanikovo se rješenje može iščitati iz podataka predanih kroz parametar data.

`String encodeText(String data);` Svrha ove metode je transformiranje cijelog predanog niza znakova u niz koji se sigurno može prikazati u ciljnoj tehnologiji.
`String encodeText(String data, int start, int length);` Svrha ove metode je transformiranje segmenta predanog niza znakova u niz koji se sigurno može prikazati u ciljnoj tehnologiji.

TestRenderer odnosno prikazivač provjere znanja jest komponenta zadužena za prikazivanje cijele provjere znanja (odnosno onog dijela koji ispitanik treba vidjeti u određenom trenutku). Ovo obuhvaća posao razmještanja zadataka po stranici (engl. layout composition), pri čemu o prikazu samih zadataka brinu odgovarajući prikazivači zadataka. Prikazivač provjere znanja također je direktno vezan uz tehnologiju prikaza, što znači da je za više tehnologija koje se žele podržati potrebno napisati više prikazivača provjere znanja, kao i za svaku posebnu tehnologiju odgovarajući skup prikazivača zadataka. TestRenderer propisuje dvije obavezne metode koje moraju biti implementirane (Tablica 4-7).

Tablica 4-7. Metode prikazivača provjere znanja

`void renderTest(`
`  ProblemInstance[] problemInstances,`
`  int startIndex,  String[] actions,`
`  String purpose,  Properties properties,`
`  List variables);`
Metoda čija je uloga pripremiti prikaz same provjere znanja. Provjeru treba prikazati u određenu svrhu (rješavanje ili samo čitanje), pri čemu treba uključiti zadatke iz polja problemInstances, i to počev od pozicije startIndex. Popis akcija koje se treba ponuditi korisniku su u polju actions. Prikaz se obavlja postavljanjem elemenata u zadnja dva parametra (properties i variables).

`void extractSolutions(`
`  ProblemInstance[] problemInstances,`
`  int startIndex,`
`  Properties properties,`
`  List variables);`
Iz predanih parametara (properties i variables) metoda očitava što je ispitanik upisao i ažurira odgovarajuće primjerke zadataka, predane u polju problemInstances, počev od pozicije startIndex.

ProblemGenerator odnosno generator zadatka je komponenta koja čuva osnovne informacije o samom zadatku, poput jedinstvenog identifikatora zadatka, popisa stvaratelja i vrednovatelja i sl.

Tablica 4-8. Elementi koncepta ProblemGenerator

ProblemGenerator		
Tip elementa	Naziv elementa	Opis elementa
String	generatorPN	javno jedinstveno ime generatora; tipično URI koji jednoznačno opisuje dotični generator
ProblemType	problemType	tip zadatka
Boolean	autoGradingSupported	podržava li ovaj generator automatsko vrednovanje ispitanikovih odgovora; postavljeno na true ako da, false inače

ProblemEditor odnosno uređivač zadatka je komponenta koja omogućava definiranje parametara koje nudi generator zadataka, čime se stvaraju novi zadaci (koncept Problem). Primjerice, generator zadatka može ponuditi stvaranje zadatka oblika: "Pronađite minimalni oblik funkcije f(A,B,...) = suma minterma (slučajni popis).", pri čemu je broj varijabli funkcije određen uniformnom distribucijom nad skupom [avg-offset, avg+offset]. Uređivač zadatka će tada ponuditi specijalizaciju parametara avg i offset, i time stvoriti jedan zadatak (primjerice, sve funkcije bit će funkcije od 4 varijable, što se postiže odabirom avg=4, offset=0). Ovaj je zadatak potom moguće uključiti u provjeru znanja. ProblemEditor propisuje dvije obavezne metode koje moraju biti implementirane, Tablica 4-9.

Tablica 4-9. Metode uređivača zadatka

`HashMap getEditorData(`
`  ProblemGenerator problemGenerator,`
`  Problem problem,`
`  HashMap properties);`
Metoda koja za zadani generator zadatka i sam zadatak a na temelju dodatnih podataka dostupnih kroz tablicu raspršenog adresiranja vraća novu tablicu raspršenog adresiranja koja sadrži podatke o prikazu uređivača zadatka u ciljnoj tehnologiji.

`ErrorStr getProblemFromData(`
`  HashMap data,`
`  Problem problem);`
Metoda koja na temelju podataka karakterističnih za ciljnu tehnologiju čita parametre zadatka i pohranjuje ih u predani zadatak.

Problem odnosno zadatak predstavlja jednu konkretnu parametrizaciju generatora zadatka. Time se provjera znanja definira kao skup zadataka koje ona uključuje. Ovdje je potrebno skrenuti pažnju na činjenicu da Problem ne predstavlja samo pitanje koje student dobiva. Naime, parametrizacijom generatora zadatka još uvijek nije stvoreno samo pitanje (u našem primjeru odabrana neka funkcija od 4 varijable).

Tablica 4-10. Elementi koncepta Problem

Problem		
Tip elementa	Naziv elementa	Opis elementa
IRepositories	repositories	repozitorij u koji se pohranjuju svi podaci specifični za ovaj zadatak
ProblemGenerator	problemGenerator	kojem generatoru pripada ovaj zadatak

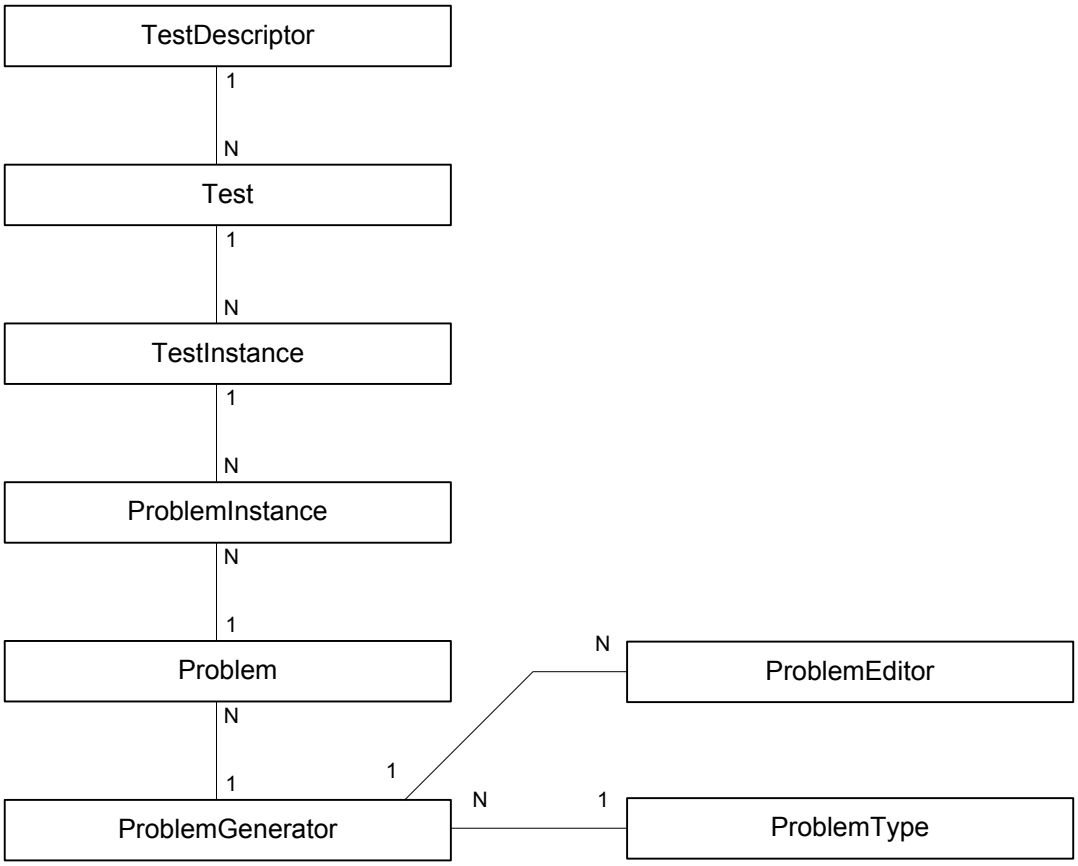
ProblemInstance odnosno primjerak zadatka predstavlja jedno konkretno pitanje koje se nudi ispitaniku na rješavanje. Prilikom stvaranja primjerka zadatka na temelju parametara definiranih u zadatku obavlja se i zadnji korak odabira vrijednosti za sve slučajne varijable, što u našem primjeru znači odabir minterma koji čine funkciju koju treba minimizirati. Budući da se ovaj korak obavlja za svakog ispitanika posebno, vrlo je mala vjerojatnost da će dva studenta dobiti iste funkcije (u

našem slučaju gdje slučajno biramo funkciju od 4 varijable vjerojatnost odabira iste funkcije iznosi 1/65536≈0,0000153).

Tablica 4-11. Elementi koncepta ProblemInstance

ProblemInstance		
Tip elementa	Naziv elementa	Opis elementa
IRepositories	repositories	repozitorij u koji se pohranjuju svi podaci specifični za ovaj primjerak zadatka
Problem	problem	definira kojeg je zadatka ovo primjerak
Boolean	problemSolved	je li zadatak rješavan od strane ispitanika; za potvrdan odgovor True, false inače
Double	correctnessMeasure	mjera točnosti zadatka; to je decimalni broj iz intervala [0,1] gdje 0 znači apsolutno netočno, a 1 apsolutno točno
Double	confidencyMeasure	mjera pouzdanosti ispitanika u uneseno rješenje; to je decimalni broj iz intervala [0, 1], gdje 0 znači da je ispitanik maksimalno nesiguran u rješenje, a 1 da je maksimalno siguran
Integer	gradingStatus	status bodovanja ovog zadatka; Tablica 10-3 definira skup prihvatljivih vrijednosti
Integer	evalStatus	status vrednovanja ovog zadatka; Tablica 10-4 definira skup prihvatljivih vrijednosti
Double	score	broj bodova zadatka
Double	maxScore	maksimalni broj bodova koji bi zadatak dobio da je bio apsolutno točan
Integer	problemInstantiationStatus	status stvaranja primjerka zadatka; Tablica 10-5 definira skup prihvatljivih vrijednosti

Slika 4-8 grafički prikazuje odnose između dijela do sada obrađenih objekata.



Slika 4-8. Odnosi između dijela osnovnih objekata modela

ProblemInstantiator odnosno stvaratelj primjerka zadatka je komponenta zadužena za stvaranje primjerka zadatka na temelju parametriziranja generatora zadatka u obliku zadatka. ProblemInstantiator propisuje jednu obaveznu metodu koja mora biti implementirana (Tablica 4-12).

Tablica 4-12. Metode stvaratelja primjerka zadatka

```
ProblemInstance instantiateProblem(
  Problem problem,
  TestInstance testInstance);
```

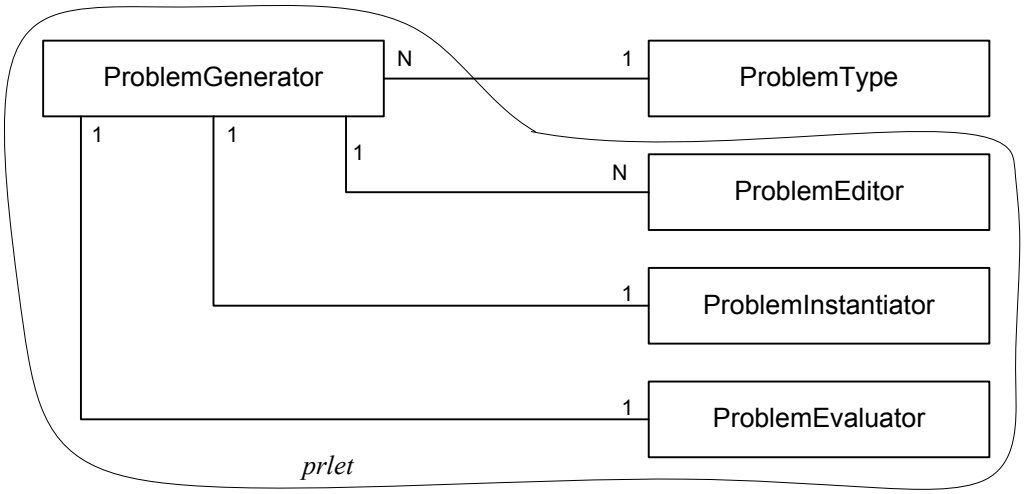
Zadatak ove metode je stvaranje jednog konkretnog primjerka zadatka. Argumenti metode su sam zadatak te primjerak testa kojemu će zadatak pripadati.

ProblemEvaluator je komponenta zadužena za vrednovanje točnosti rješenja primjerka zadatka. Rezultat procesa vrednovanja odgovora je decimalni broj iz intervala [0, 1], pri čemu vrijednost 0 znači da je odgovor apsolutno pogrešan, a vrijednost 1 apsolutno točan. ProblemEvaluator propisuje jednu obaveznu metodu koja mora biti implementirana (Tablica 4-13). Ovisno o izvedbi vrednovatelja zadatka, u metodu evaluateProblemInstance osim samog vrednovanja unesenog rješenja pojedini vrednovatelji mogu obaviti i proces stvaranja točnog rješenja.

Tablica 4-13. Metode vrednovatelja primjerka zadatka

```
boolean evaluateProblemInstance(
  ProblemInstance problemInstance);
```
Zadatak ove metode jest vrednovanje rješenja ispitanika, te stvaranje točnog rješenja ukoliko je to podržano i ispitanikovo rješenje nije točno. Metoda vraća true u slučaju uspjeha, false inače.

Slika 4-9 prikazuje građu prleta, uzevši u obzir do sada definirane koncepte.



Slika 4-9. Građa prleta

Kako bi model podržao istodobno omogućavanje početka pisanja više provjera znanja, model definira još dva koncepta: red za omogućavanje provjera znanja, te zahtjev za omogućavanje početka pisanja.

TestStartQueue predstavlja red za omogućavanje provjera znanja. Uporaba ovog reda detaljno će biti objašnjena u sljedećem poglavlju. Tablica 4-14 prikazuje elemente ovog reda.

Tablica 4-14. Elementi koncepta TestStartQueue

TestStartQueue		
Tip elementa	Naziv elemenata	Opis elemenata
String	name	naziv reda
String	description	opis reda
User	owner	vlasnik reda
Set<TestStartRequest>	requests	skup zahtjeva za omogućavanjem
IRepositories	repositories	repozitorij za pohranu privatnih podataka reda

TestStartRequest predstavlja jedan konkretan zahtjev za omogućavanjem početka pisanja primjerka provjere znanja jednog studenta. Tablica 4-15 prikazuje elemente povezane s ovim konceptom koji su uz prethodni opis sasvim jasni.

Tablica 4-15. Elementi koncepta TestStartRequest

TestStartRequest		
Tip elementa	Naziv elemenata	Opis elemenata
TestStartQueue	testStartQueue	red kojem zahtjev pripada
TestInstance	testInstance	primjerak provjere znanja koji se želi omogućiti
Integer	status	status ovog zahtjeva; za sada je rezerviran i mora biti 0
IRepositories	repositories	repozitorij za pohranu privatnih podataka ovog zahtjeva

Uz do sada opisane koncepte, potrebno je definirati i koncepte koji određuju bodovanje primjeraka zadataka, nakon što se utvrdi mjera točnosti.

Grader je komponenta koja je zadužena za dodjelu bodova primjerku zadatka. Odluka o tome hoće li svi zadaci unutar jednog primjerka provjere znanja nositi isti broj bodova ili ne spada u politiku kolegija, što znači da ova komponenta mora podržati različite načine dodjele bodova, u ovisnosti o statusu rješavanosti primjerka zadatka, mjeri točnosti zadatka i mjeri pouzdanosti ispitanika. Ova komponenta spada u komponente s mogućnošću podešavanja, što znači da osim metoda namijenjenih bodovanju (Tablica 4-16) mora imati i metode za podešavanje parametara (Tablica 4-17).

Tablica 4-16. Metode ocjenjivača primjerka zadatka

```
public void grade( ProblemInstance problemInstance,
  TestInstance testInstance);
```
Zadatak ove metode jest dodjela bodova primjerku zadatka u skladu s podešenom politikom bodovanja.

Tablica 4-17. Metode podesivih komponenti

```
public List<ParamDescription> listRequiredParams(
  String purpose);
```
Metoda vraća popis parametara (naziv, kratki naziv, opis, tip, je li obavezan) koji se mogu podešavati kod ove podesive komponente.

```
public List<ParamErrorDescription> checkParams(
  String purpose,
  IRepository repository);
```
Metoda provjerava imaju li svi parametri ove komponente legalne vrijednosti, a ako ih nemaju, ona vraća popis pronađenih pogrešaka.

Vezano uz sam proces pisanje provjere znanja, definiraju se još tri koncepta: TestStartChecker, TestSupervisor te TestController.

TestStartChecker odnosno provjerivač mogućnosti pokretanja provjere znanja je komponenta čija je zadaća provjeriti ima li ispitanik sve potrebne dozvole i preduvjete za pristup samoj provjeri znanja. Ovo je prva vrsta komponenti koja će omogućiti definiranje politike kolegija i njezino provođenje. Tablica 4-18 prikazuje popis metoda koje ova komponenta mora implementirati.

Tablica 4-18. Metode provjerivača dozvole početka pisanja provjere znanja

`boolean isStartAllowed(`
`  ITestStartCheckerSupport support,`
`  Properties params,`
`  List explanations);`
Zadatak ove metode jest provjeriti jesu li svi preduvjeti koje propisuje ovaj provjerivač zadovoljeni, kako bi se moglo pristupiti provjeri znanja. Metoda vraća true ako jesu, false inače.

`Results enableTest(`
`  IRepositories tdRepositories,`
`  IRepositories tiRepositories,`
`  IRepository qRepository);`
Zadatak ove metode jest omogućiti pristup provjeri znanja. Metoda će tipično biti pozvana kao rezultat intervencije osobe koja nadzire postupak provođenja provjere znanja.

`boolean initializeParameters(`
`  ITestStartCheckerSupport support,`
`  Properties client_params);`
Metoda služi za inicijalizaciju parametara samog provjerivača. Metoda se poziva samo jednom.

Provjerivač mogućnosti pokretanja provjere znanja oslanja se na uslugu sustava koja će mu ponuditi sve potrebne informacije i metode za manipulaciju nad samim primjerkom provjere znanja. Objekt koji to nudi jest ITestStartCheckerSupport. Tablica 10-6 definira popis metoda koje taj objekt nudi na uporabu.

TestSupervisor ili nadglednik (tijeka pisanja) provjere znanja je komponenta čija je zadaća nadzor nad tijekom pisanja provjere znanja. Za razliku od komponente TestStartChecker koja se konzultira samo na početku pisanja provjere znanja, ova komponenta konzultira se prilikom svakog pokušaja interakcije s primjerkom provjere znanja, i to prije no što se uvaži akcija koju je ispitanik zatražio. Osim toga, prilikom interakcije s ovom komponentom, ona je u mogućnosti generirati različite objave (engl. announcements) koje će potom biti prikazane ispitaniku, ili će se na temelju tih objava poduzeti odgovarajuće akcije u korisničkom sučelju samog ispitanika. Tablica 4-19 prikazuje popis metoda koje ova komponenta mora implementirati.

Tablica 4-19. Metode nadglednika tijeka pisanja provjere znanja

`void testStarted(ITestSupervisorSupport support);` Metoda se poziva u trenutku kada se korisniku omogući pristup početku pisanja provjere znanje, kako bi nadglednik doznao točan trenutak početka i obavio potrebnu inicijalizaciju svojih parametara.

`Results checkTestInterractionAllowed(`
`  ITestSupervisorSupport support);`
Zadaća ove metode je provjeriti smije li korisnik još uvijek obavljati interakciju s provjerom znanja. Ukoliko ne smije, vraća odgovarajuća obrazloženja kao povratnu vrijednost.

`void fillInfo(`
`  ITestSupervisorSupport support,`
`  Properties properties);`
Zadaća ove metode je prihvaćanje objava koje nadglednik stvara za ispitanika i koje će na odgovarajući način biti prikazane ispitaniku.

Nadglednik tijeka pisanja provjere znanja za svoj rad također se oslanja na usluge koje mu mora ponuditi sam sustav. Sustav te usluge mora ponuditi u obliku objekta ITestSupervisorSupport. Tablica 10-7 definira popis metoda koje taj objekt nudi na uporabu.

TestController ili upravljač tijeka pisanja provjere znanja spada u jednu od najvažnijih komponenti, jer direktno utječe na izvedbu samog procesa pisanja provjere znanja. To je komponenta čija je uloga odabrati zadatke koji će sudjelovati u provjeri znanja, definirati koje akcije korisnik može obaviti tijekom pisanja te provjere (primjerice, je li dozvoljena uporaba tipke "natrag" (engl. back)) i sl. Zahvaljujući izdvajanju ove funkcionalnosti u zasebnu komponentu, u sustav koji je u skladu s ovim modelom izuzetno se lagano mogu dodavati novi načini vođenja ispitnog procesa (prilagodljivi testovi, inteligentno vođeni testovi i sl.). Tablica 4-20 prikazuje metode koje mora implementirati upravljač tijeka pisanja provjere znanja.

Kao temeljnu strukturu za pohranu privatnih podataka mnogi objekti modela koriste koncept repozitorija podataka. U kontekstu modela StudTest, stoga se definiraju sljedeći objekti: IRepositories (Tablica 10-9), IRepository (Tablica 10-10), IKeyRepository (Tablica 10-11), IAttachmentRepository (Tablica 10-12), te IAttachment (Tablica 10-13). Pri tome IRepositories predstavlja kolekciju repozitorija jednoznačno definiranih svojim imenom, IRepository predstavlja jedan konkretan repozitorij koji se sastoji od dva dijela – repozitorija jednostavnih vrijednosti te repozitorija privitaka. Repozitorij jednostavnih vrijednosti (IKeyRepozitory) pohranjuje niz vrijednosti koje su jednoznačno određene svojim imenom (čime zapravo čuva skup parova ime=vrijednost). Repozitorij privitaka pohranjuje niz privitaka (IAttachment), pri čemu je svaki privitak također jednoznačno određen svojim imenom. Zahvaljujući ovim strukturama, dodavanje statičkih višemedijskih sadržaja bit će znatno olakšano, dok će istovremeno sve komponente koje trebaju pohranjivati privatne podatke to moći činiti na uniforman način, pri čemu se odgovornost za način pohrane podataka (u datoteke, u bazu podataka i sl) izolira u potpornu infrastrukturu nad kojom sve komponente žive.

Tablica 4-20. Metode upravljača tijeka pisanja provjere znanja

`Results prepareTestInstance(`
`  ITestControllerSupport support)`
`  throws DataStorageException;`
Metoda koja se poziva u trenutku stvaranja primjerka provjere znanja radi osiguravanja neprilagodljivim izvedbama upravljača mogućnost odabira i stvaranja primjeraka svih zadataka koji će biti uključeni u primjerak provjere znanja.

`boolean isMultivisitAllowed(`
`  TestInstance testInstance,`
`  IRepositories tdRepositories);`
Metoda čiji je zadatak utvrditi je li dozvoljen višestruki pristup primjerku provjere znanja. Ako jest, metoda vraća true, inače false.

`void onNewVisit(`
`  TestInstance testInstance,`
`  IRepositories repositories);`
Metoda koja će biti pozvana u trenutku ponovnog pristupa primjerku provjere znanja kako bi se upravljač obavijestio o tom događaju (ako je to dozvoljeno).

`boolean solutionStorageAllowed(`
`  TestInstance testInstance,`
`  IRepositories repositories,`
`  String action);`
Zadatak ove metode je provjeriti je li pohrana odgovora koje je ispitanik unio dozvoljena u trenutnom stanju primjerka provjere znanja. Pri tome se kao jedan od argumenata predaje i trenutna akcija koju je korisnik odabrao.

`int onAction(`
`  TestInstance testInstance,`
`  IRepositories repositories,`
`  String action);`
Metoda koja određuje što se sljedeće mora dogoditi ako je korisnik odabrao predanu akciju. Povratna vrijednost mora biti jedna od konstanti (Tablica 10-8) čime se definira njegovo nastavljanje, zamrzavanje ili završetak provjere znanja (koji se u tom slučaju može poslati na ocjenjivanje).

`int[] getTaskRangeToDisplay(`
`  TestInstance testInstance,`
`  IRepositories repositories);`
Metoda utvrđuje raspon primjeraka zadataka koje kao sljedeće treba prikazati ispitaniku.

`String[] getActions(`
`  TestInstance testInstance,`
`  IRepositories repositories);`
Metoda vraća sve akcije koje u aktualnom trenutku korisnik može poduzeti a bit će ispitaniku ponuđene kroz odgovarajuće korisničko sučelje, zajedno s samim zadacima.

`void calculateAndSetScore(`
`  TestInstance testInstance,`
`  IRepositories repositories);`
Metoda računa ukupni broj bodova primjerka provjere znanja i utvrđuje je li provjera položena ili ne.

4.5 Komponente za potporu dinamičkoj podršci izrade višemedijskih sadržaja

Kako bi se osigurala podrška zadacima za uporabu višemedijskih sadržaja koji se stvaraju u letu (na zahtjev) na temelju parametara pohranjenih u primjerak zadatka, potrebno je definirati još jednu komponentu.

HelperWorker odnosno pomoćnik je komponenta koja na zahtjev stvara sliku, zvuk i slične sadržaje. Pri tome pomoćnik dobiva potrebne parametre iz repozitorija primjerka zadatka, i to u obliku parova *ime=vrijednost*, koji će se predavati u obliku strukture mape. Za stvoreni sadržaj pomoćnik vraća: (i) sam sadržaj u obliku polja okteta, (ii) MIME tip stvorenog sadržaja [28], (iii) zastavicu je li stvoreni sadržaj statički ili će svaki puta biti drugačiji, unatoč istim parametrima iz repozitorija (što određuje može li se takav sadržaj pohraniti u priručnu memoriju – engl. cache) te (iv) mapu dodatnih parametara koji su specifični za svaku vrstu stvorenog sadržaja. Kroz ovu mapu pomoćnik će slati informacije poput širine i visine, ukoliko je rezultat rada pomoćnika slika, ili pak vremensko trajanje ukoliko je u pitanju zvučni zapis i sl.

Tablica 4-21 definira metode koje svaki pomoćnik mora implementirati.

Tablica 4-21. Metode pomoćnika

`HelperWorkerResultInfo getContentInfo(Map data);` Na temelju parametara predanih kroz mapu data metoda utvrđuje sve parametre sadržaja koji bi bio stvoren za na temelju predanih parametara (osim samog sadržaja). Ova metoda služi brzom uvidu u karakteristike stvorenog sadržaja, bez potrebe za stvaranjem samog sadržaja.
`HelperWorkerResult getContent(Map dana);` Na temelju parametara predanih kroz mapu data metoda utvrđuje sve parametre sadržaja te stvara i sam sadržaj za predane parametre.

Upravo zahvaljujući konceptu pomoćnika ovaj model omogućava stvaranje dinamičkih višemedijskih sadržaja koji se tada mogu koristiti za izradu bogatih i složenih zadataka, a bez da se pri tome sami zadaci opterećuju implementacijom odgovarajuće funkcionalnosti. Osim toga, ovaj pristup omogućava dijeljenje funkcionalnosti pomoćnika između više zadataka te su optimalan pristup za ispitivanje u okviru kolegija, gdje se može analizirati kakve sve primjerice slike treba generirati i potom napisati odgovarajući skup pomoćnika koji će se potom dijeliti između svih zadatka kolegija.

4.6 Potpora tehnologijama Semantičkog Weba

Jedan od trendova današnjice jest nagli razvoj tehnologija i vizije Semantičkog Weba [46][47][48]. Ideja Semantičkog Weba je dati svemu što se pojavljuje na Webu značenje koje strojevi mogu razumjeti, te na temelju toga obavljati zaključivanje, što bi u konačnici imalo pružanje pomoći zadacima koje trenutno rade isključivo ljudi. Osnovni preduvjet uporabe ovih tehnologija jest uvođenje mehanizma kojim se izbjegavaju višeznačnice – svaki pojam o kojem se govori mora imati svoje jedinstveno ime, koje se u kontekstu Semantičkog Weba naziva URI. U kontekstu e-

ispitivanja ovo podrazumijeva opisivanje studentskih provjera znanja, zadataka, postignutog broja bodova i sl.

Uz sve do ovog trenutka opisane komponente, radi potpore uporabi tehnologija Semantičkog Weba model StudTest definira dodatni uvjet:

> *svi prleti, zadaci te opisnici provjera znanja moraju biti opisani jedinstvenim identifikatorom, čime će se osigurati mogućnost njihovog jednoznačnog referiranja s bilo kojeg mjesta.*

Ovaj je uvjet nužan kako bi model pružio potporu uključivanju ovih objekata u opise korištene za algoritme umjetne inteligencije. Jedna primjena ovih identifikatora bit će predstavljena u poglavlju o inteligentnom vođenju ispitnog procesa.

5 Model interakcije komponenti sustava e-ispitivanja StudTest

U prethodnom poglavlju definirani su osnovni koncepti s kojima je sagrađen cjelokupan model StudTest. Međutim, radi daljnjeg definiranja funkcija pojedinih komponenti kao i načina njihova ispunjavanja, potrebno je još definirati i skup osnovnih interakcija. U postupku upravljanja provjerama znanja mogu se razlučiti sljedeći općeniti zadaci: (i) stvaranje novog opisnika provjere znanja i uređivanje parametara te (ii) pisanje provjere znanja. Ovi se zadaci obrađuju u nastavku.

5.1 Stvaranje novog opisnika provjere znanja

Prvi korak u radu s provjerama znanja je stvaranje nove provjere znanja. Od konkretne implementacije ovog modela očekuje se rješavanje problema pohrane podatkovnih struktura koje odgovaraju opisniku provjere znanja. Međutim, u trenutku stvaranja opisnika propisuje se da repozitorij opisnika mora biti prazan. Ukoliko se u nekoj konkretnoj objektno orijentiranoj implementaciji modela za pohranu koristi relacijska baza podataka, ovaj korak odgovara sljedećim aktivnostima:

- stvori novi primjerak razreda TestDescriptor,
- osiguraj da je repozitorij primjerka prazan,
- na temelju opisa razreda obavi umetanje novog retka u relacijsku bazu podataka pri čemu se u odgovarajuće stupce relacije pohranjuju svi elementi primjerka.

Nakon ovoga, opisnik je spreman za podešavanje. Podešavanje se svodi na definiranje nekoliko parametara (Tablica 5-1). Pri tome se za upravljač, ocjenjivač te liste provjerivača i nadglednika u repozitorij upisuju imena implementacija odgovarajućih komponenti. Pri tome se liste imena upisuju kao slijed zarezima odvojenih elemenata.

Tablica 5-1. Parametri komponente TestDescriptor

Parametar	Mjesto pohrane
naziv	u element koncepta TestDescriptor
opis	u element koncepta TestDescriptor
upravljač	u repozitorij "generalData", ključ "graderName"
ocjenjivač	u repozitorij "generalData", ključ "controllerName"
politika stvaranja	u element koncepta TestDescriptor
popis provjerivača	u repozitorij "generalData", ključ "startCheckersList"
popis nadglednika	u repozitorij "generalData", ključ "supervisorsList"

Slika 5-1 prikazuje primjer sučelja za uređivanje ovih parametara. Odabir provjerivača i nadzornika u ovom se primjeru obavlja kroz posebne stranice, jer se u sustavu očekuje velik broj provjerivača i nadglednika. Nakon definiranja ovih komponenti, potrebno je obaviti njihovu konfiguraciju.

Slika 5-1. Primjer sučelja za podešavanje parametara opisnika provjere znanja

5.2 Konfiguriranje podesivih komponenti

Nakon što su odabrane podesive komponente koje se želi koristiti za provjeru znanja, potrebno je obaviti njihovu konfiguraciju. Podesive komponente povezane s opisnikom provjere znanja su redom: upravljač, ocjenjivač, provjerivači te nadglednici. Ovaj postupak provodi se u tri koraka, neovisno o vrsti komponente.

U prvom koraku za odabranu podesivu komponentu poziva se metoda `listRequiredParams("forTestDescriptor")` koja vraća listu parametara koje je potrebno ponuditi korisniku na uređivanje. Zatim se nad repozitorijom općih podataka za trenutno pohranjene parametre poziva metoda `checkParams("forTestDescriptor",repository("generalData"))` koja će provjeriti vrijednosti definirane za te parametre i generirati popis pogrešaka (ako postoje; ovo neće biti slučaj kod prvog uređivanja kada vrijednosti još nisu definirane).

U drugom koraku korisniku se parametri nude na uređivanje, te se ispisuju pogreške uz one parametre za koje je komponenta utvrdila da nisu valjani.

U trećem koraku parametri se pohranjuju u repozitorij "generalData" opisnika testa pri čemu se parametri podesive komponente upisuju kao ključevi čija vrijednost odgovara imenu parametra i koji se povezuju na vrijednost koja odgovara vrijednosti parametra.

Više detalja o samom podešavanju ovih komponenti ovdje se ne može razraditi jer model ne propisuje niti jednu obaveznu implementaciju neke od komponenti. Umjesto toga, ovdje opisani postupak je potpuno općenit, te će primjeri stvarno implementiranih komponenti biti prikazani u posebnom poglavlju.

5.3 Poslovanje provjerom znanja

Poslovanje provjerom znanja jedan je od najsloženijih zadataka sustava. Sve implementacije ovog modela dužne su obaviti stvaranje novih praznih primjeraka provjera (način nije propisan). Prazna provjera znanja jest provjera znanja kojoj je repozitorij prazan, i koja ima postavljen status Jednom kada je prazna provjera stvorena, prolazi se kroz sljedeći niz koraka:

- početno stvaranje provjere znanja,
- pisanje provjere znanja,
- ocjenjivanje provjere znanja.

Svaki od ovih koraka opisan je u nastavku.

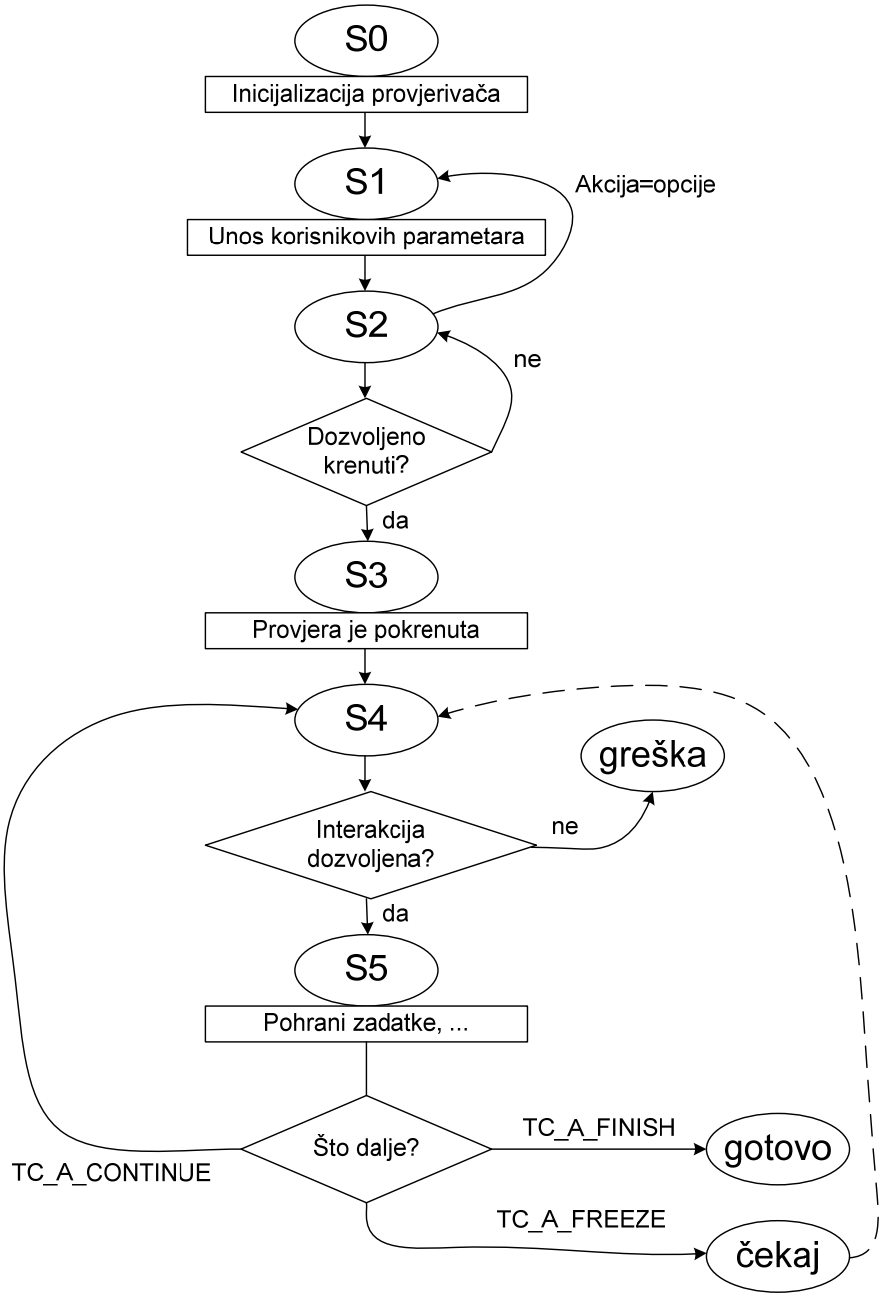
5.3.1 Početno stvaranje provjere znanja

Nakon što je stvorena prazna provjera znanja status primjerka je TST_UNSOLVED. Iz repozitorija generalData opisnika provjere znanja dohvaća se naziv podešenog upravljača provjere te se na temelju tog naziva dohvaća odgovarajuća komponenta. Status provjere prebacuje se u TST_PREPARING_IN_PROGRESS. Nad upravljačem se poziva metoda `prepareTestInstance`. Nakon što odabrani upravljač obavi pripremu provjere znanja, status provjere se prebacuje u TST_PREPARED.

Obim poslova koje treba obaviti unutar metode `prepareTestInstance` ovisi o samom upravljaču. Primjerice, upravljač koji poslužuje statičke provjere znanja ovu će metodu iskoristiti za odabir i stvaranje svih primjeraka zadataka koji će biti uključeni u provjeru. Ako je riječ o upravljaču koji poslužuje prilagodljive provjere znanja, tada se ova metoda može iskoristiti za odabir i stvaranje početnih primjeraka zadataka koji će prvi biti posluženi korisniku.

5.3.2 Pisanje provjere znanja

Pisanje provjere znanja započinje u trenutku kada korisnik zatraži primjerak provjere znanja čiji je status TST_PREPARED. Postupak je sljedeći. Dohvaća se komponenta upravljač provjere znanja. Iz repozitorija se dohvaća vrijednost Progress koja je pohranjena kao ključ "hr.fer.zemris.studtest.core.testWriters.TestWriter#Progress". Ova vrijednost predstavlja cijeli broj čija je uloga praćenje redoslijeda akcija koje izvršava korisnik. Ovime se dolazi do konačnog automata koji je zadužen za upravljanje tijekom provjere znanja, a prikazan je u obliku dijagrama toka stroja stanja (Slika 5-2).



Slika 5-2. Upravljanje tijekom provjere znanja

Ulaskom u stanje S0 iz repozitorija se dohvaća lista svih provjerivača te se pozivom metode `initializeParameters` obavlja njihova inicijalizacija. Potom se bezuvjetno prelazi u stanje S1.

Ulaskom u stanje S1, provjerava se zahtijevaju li provjerivači kakve parametre od ispitanika (pozivom metode `listRequiredParams`, uz argument "forTestWriting"). Ukoliko postoje zahtjevi, korisniku se nudi mogućnost njihovog unosa. Prelazi se u stanje S2.

Ulaskom u stanje S2 dohvaća se lista svih provjerivača, i nad njima redom poziva metoda `isStartAllowed`. Ukoliko barem jedan od provjerivača vrati false, pristup provjeri se odbija, i vraća se u stanje S2. U suprotnom, prelazi se u stanje S3.

Ulaskom u stanje S3 dohvaća se lista nadglednika, te se nad njima poziva metoda `testStarted`. Status provjere znanja također se mijenja i prelazi u TST_SOLVING_IN_PROGRESS. Prelazi se u stanje S4.

Ulaskom u stanje S4 dohvaća se lista nadglednika, te se nad njima poziva metoda `checkTestInterractionAllowed`, kako bi se provjerilo smije li ispitanik uopće obavljati ikakve izmjene nad primjerkom provjere znanja. Ako bilo koji nadglednik vrati false, primjerak provjere znanja se poništava (naime, korisnik pokušava mijenjati provjeru, a nadglednici koji su podešeni u skladu s politikom kolegija to ne dopuštaju). Ako je sve u redu, prelazi se u stanje S5.

Ulaskom u stanje S5 obavlja se čitav niz akcija. Najprije se poziva metoda `solutionStorageAllowed` upravljača. Ukoliko upravljač dopusti pohranu rješenja, analiziraju se poslana rješenja i pohranjuju u odgovarajuće primjerke zadataka. Ako pohrana nije dopuštena, bit će preskočena (jer je i to sasvim legalna akcija). Potom se poziva metoda `onAction` nad upravljačem, koja će definirati kako dalje nastaviti s provjerom znanja. Moguća su tri odgovora: završi provjeru, zamrzni provjeru te nastavi dalje. U slučaju prvoga status provjere se postavlja TST_SOLVING_FINISHED, i provjera se šalje na ocjenjivanje. U slučaju drugoga, pisanje se privremeno zamrzava, i bit će ga moguće nastaviti kada to ispitanik odluči. U slučaju trećega, nastavlja se provjera znanja, te se od svih nadglednika zove metoda `fillInfo`, kako bi se pokupile sve objave koje dotični stvaraju. Potom se od upravljača poziva `getTaskRangeToDisplay` kako bi se dobio popis zadataka koje sljedeće treba prikazati. Slijedi poziv metode `getActions` (također nad upravljačem) kako bi se dobio popis svih akcija koje treba ponuditi ispitaniku (akcije mogu biti bilo što što upravljač definira, a najčešće su "Next" i "Finish test"). Na kraju, pronalazi se odgovarajući prikazivač testa i obavlja postupak definiranja prikaza (metodom `renderTest`), te se vrijednost ključa Progress uvećava za 1. Stanje automata prebacuje se u S4, i korisniku se prikazuju zadaci.

5.3.3 Ocjenjivanje provjere znanja

U trenutku kada se završi pisanje provjere znanje (a kod prilagodljivih provjera i prije) potrebno je ocijeniti provjeru znanja. U tu svrhu potrebno je najprije obaviti postupak vrednovanja svih nevrednovanih zadataka i potom obaviti konačno bodovanje provjere.

Postupak vrednovanja zadatka provodi se tako da se za svaki primjerak zadatka utvrdi kojem generatoru on pripada, i na temelju te informacije pronađe odgovarajući ProblemEvaluator. Zadatak se vrednuje pozivom metode `evaluateProblemInstance` odgovarajuće komponente ProblemEvaluator.

Nakon što su svi zadaci vrednovani (te im je utvrđena mjera točnosti), za svaki se zadatak uporabom ocjenjivača podešenog u opisniku provjere metodom `grade` računa broj bodova (zapravo, ocjenjivač postavlja dva polja u primjerku zadatka: dobiveni broj bodova, te maksimalno mogući broj bodova koji odgovara broju bodova uz pretpostavku da je mjera točnosti 1).

Konačno, nakon što je obavljeno vrednovanje i bodovanje svih zadataka, potrebno je odrediti je li provjera položena ili nije. U tu svrhu ponovno se pronalazi upravljač provjere znanja, i poziva njegova metoda `calculateAndSetScore`, koja će

obaviti taj posao. Primjerice, jedna moguća implementacija ove metode može uzeti u obzir postignuti broj bodova te maksimalno mogući broj bodova (te podatke za svaki je primjerak zadatka ocjenjivač izračunao i pohranio) i na temelju toga je li test položen ili nije uzimajući u obzir neki unaprijed definirani prag.

5.4 Zahtijevanje pokretanja testa

Kako bi ponudio prikladni nadzor nad uvjetima u kojima se obavlja pisanje provjera znanja, model predviđa postojanje komponente TestStartQueue, koja će omogućiti redanje zahtjeva korisnika za davanjem dozvole za pristup provjeri znanja, te davanje dozvole u trenutku kada to odluči osoba zadužena za nadzor provjere. Da bi to bilo moguće, potrebno je definirati i jedan provjerivač koji ćemo nazvati TestStartQueueChecker, i taj provjerivač mora biti pridružen opisniku provjere znanja. U tom slučaju, događa se sljedeće. Prilikom ulaska u stanje S1, provjerivač će objaviti da korisnik treba upisati naziv reda za omogućavanje provjera. Nakon što to korisnik unese, ulaskom u stanje S2 i pozivom metode `isStartAllowed`, provjerivač će u definirani red poslati zahtjev za omogućavanjem provjere, i vratiti false.

Paralelno, osoba zadužena za nadzor provođenja provjere znanja (primjerice asistent) spojit će se na isti red (kroz administracijsko sučelje) i omogućiti pisanje zahtjevima koje vidi u redu – što će za posljedicu imati poziv metode `enableTest` nad našim provjerivačem.

Sljedećim pokušajem pristupa provjeri znanja, `isStartAllowed` provjerivača vratit će true te će ispitanik moći pristupiti provjeri znanja.

5.5 Stvaranje novog primjerka zadatka na temelju zadatka

Stvaranje novog primjerka zadatka obavlja komponenta ProblemInstantiator. Za stvaranje novog primjerka zadatka potrebno je utvrditi kojem generatoru zadatak pripada, te na temelju te informacije utvrditi koja se komponenta unutar generatora koristi kao ProblemInstantiator. Potom se dohvaća ta komponenta, i poziva metoda instantiateProblem.

6 Potpora inteligentnom vođenju e-ispitivanja

Model sustava e-ispitivanja opisan u ovom radu pogodan je za izradu cijelog niza različitih izvedbi provjera znanja, od kojih su statičke provjere znanja najosnovnija vrsta. Zahvaljujući dobrom razdvajanju funkcionalnosti komponenti definiranih ovim modelom moguće je razmotriti kako bi sustav koji implementira predstavljeni model mogao obavljati prilagodljivo inteligentno vođenje ispitnog procesa. Kako ovaj problem zadire u srž umjetne inteligencije, te se može rješavati uporabom više tehnika umjetne inteligencije, za potrebe ilustracije poslužit ćemo se jednom od tehnika temeljenoj na uporabi opisne logike (engl. description logic) [33], odnosno semantičkim mrežama [31].

U daljnjem tekstu za potrebe prikaza znanja na temelju kojeg će se definirati adaptivno vođenje ispitnog procesa koristi se pojam *ontologija*. Prema Gruberu [29]:

Ontologija je formalna specifikacija konceptualizacije.

Ovo možda i nije najbolja definicija (primjerice, vidi raspravu o ontologijama i prikladnim definicijama [30]), no to je danas vrlo često citirana definicija i za potrebe ovog rada će poslužiti. Iako je ovdje ontologija odabrana u svrhu ilustracije načela, ontologije se u današnje doba doista sve više koriste za prikaz znanja u tehničkim sustavima (o prikazu znanja uporabom ontologija [32]).

6.1 Ontološki opis gradiva kolegija

Inteligentno vođenje ispitnog procesa razmotrit ćemo na primjeru kolegija sveučilišne razine (primjerice, Digitalne elektronike). Krenut ćemo od pretpostavke da je gradivo koje se uči opisano ontologijom O. Kao primjer ontologije možemo iskoristiti ontologiju *feronto1* [34]. U svjetlu Gruberove definicije ontologije, koncepti ove ontologije prikazani su tablično (Tablica 6-1, naveden je samo dio zanimljiv u ovom kontekstu).

Tablica 6-1. Neki koncepti ontologije feronto1

Naziv koncepta	Opis
Concept	Bilo kakav pojam koji se želi opisivati.
Course	Kolegij.
Document	Općeniti dokument.
WebPage	Podrazred dokumenta – Web stranica; URI mora predstavljati dohvativ dokument (drugim riječima, mora predstavljati URL).

Relacije iste ontologije koje vrijede između koncepata također su prikazane tablično uz kratka objašnjenja (Tablica 6-2).

Tablica 6-2. Neke relacije ontologije feronto1

Naziv relacije	Opis
conceptLabel	Definira naziv pojma. Primjer: <c:[BooleanAlgebra], rdf:[type], fer1:[Concept]> <c:[BooleanAlgebra], fer1:[conceptLabel], "Booleova algebra">
relatedTo	Definira da je neki dokument povezan s nekim konceptom. Primjer: <d:[doc1], rdf:[type], fer1:[Document]> <d:[doc1], fer1:[relatedTo], c:[BooleanAlgebra]>
relatedConcept	Definira da je neki koncept povezan s nekim drugim konceptom na neodređen način. Primjer: <c:[BooleanAlgebra], fer1:[relatedConcept], c:[Algebra]>
dependsOn	Definira da je za razumijevanje koncepta najprije potrebno razumjeti neki drugi koncept. Primjer: <c:[BooleanAlgebra], fer1:[dependsOn], c:[Algebra]>
extendsConcept	Definira da je koncept proširenje nekog drugog koncepta. Primjer: <c:[BooleanAlgebra], fer1:[extendsConcept], c:[Algebra]>
isAbout	Definira da neki dokument govori o nekom konceptu. Primjer: <d:[Algebra.html], fer1:[isAbout], c:[Algebra]>

Tablica 6-2 u primjerima koristi notaciju prikaza veza između koncepata uporabom uređenih trojki (engl. triplets), pri čemu se na prvoj poziciji nalazi početni koncept, na drugom mjestu dolazi naziv relacije a na trećem mjestu dolazi drugi koncept, koji možemo promatrati kao objekt izjave u kojoj je na prvom mjestu subjekt, a relacija predstavlja predikat. Tako primjerice izjavu:

<c:[BooleanAlgebra], fer1:[extendsConcept], c:[Algebra]>

možemo pročitati kao "koncept BooleanAlgebra proširenje je koncepta Algebra" (u smislu proširenja koje se koristi u objektno orijentiranim programskim jezicima). Također, u skladu s idejom RDF-a [35], jedne od prvih općeprihvaćenih ontologija za potporu tehnologija Semantičkog weba [36], svaki koncept predstavljen je svojim jedinstvenim identifikatorom (URI, engl. Uniform Resource Identifier), što je u primjerima zbog preglednosti zapisano kraćom sintaksom oblika pokrata:[naziv].

Tablica 6-3 definira vrijednosti pokrata korištenih u ovom poglavlju.

Tablica 6-3. Definicije korištenih pokrata kod URI-ja

Oznaka pokrate	Zamjena za
c	http://aule.zemris.fer.hr/onto/concepts#
d	http://aule.zemris.fer.hr/documents/
fer1	http://www.fer.hr/#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#

Identifikator c:[BooleanAlgebra] time je zapravo definiran kao niz znakova: "http://aule.zemris.fer.hr/onto/concepts#BooleanAlgebra".

Uporabom definirane ontologije sad se može opisati koncepte koji će biti obrađeni u okviru kolegija. Za potrebe ovog primjera definirat ćemo samo nekolicinu njih (Slika 6-1).

<c:[FlipFlops], rdf:[type], fer1:[Concept]>
<c:[FlipFlops], fer1:[conceptLabel], "Bistabili">

<d:[FlipFlops.html], rdf:[type], fer1:[WebPage]>
<d:[FlipFlops.html], fer1:[isAbout], c:[FlipFlops]>

<c:[FlipFlopTransitionTable], rdf:[type], fer1:[Concept]>
<c:[FlipFlopTransitionTable], fer1:[conceptLabel], "Tablica prijelaza bistabila">
<c:[FlipFlopTransitionTable], fer1:[dependsOn], c:[FlipFlops]>

<d:[FlipFlopTransitionTable.html], rdf:[type], fer1:[WebPage]>
<d:[FlipFlopTransitionTable.html], fer1:[isAbout], c:[FlipFlopTransitionTable]>

<c:[DFlipFlops], rdf:[type], fer1:[Concept]>
<c:[DFlipFlops], fer1:[conceptLabel], "D Bistabili">
<c:[DFlipFlops], fer1:[extendsConcept], c:[FlipFlops]>

<d:[DFlipFlops.html], rdf:[type], fer1:[WebPage]>
<d:[DFlipFlops.html], fer1:[isAbout], c:[DFlipFlops]>

<c:[TFlipFlops], rdf:[type], fer1:[Concept]>
<c:[TFlipFlops], fer1:[conceptLabel], "T Bistabili">
<c:[TFlipFlops], fer1:[extendsConcept], c:[FlipFlops]>

<d:[TFlipFlops.html], rdf:[type], fer1:[WebPage]>
<d:[TFlipFlops.html], fer1:[isAbout], c:[TFlipFlops]>

<c:[JKFlipFlops], rdf:[type], fer1:[Concept]>
<c:[JKFlipFlops], fer1:[conceptLabel], "D Bistabili">
<c:[JKFlipFlops], fer1:[extendsConcept], c:[FlipFlops]>

<d:[JKFlipFlops.html], rdf:[type], fer1:[WebPage]>
<d:[JKFlipFlops.html], fer1:[isAbout], c:[JKFlipFlops]>

Slika 6-1. Ontološki prikaz dijela koncepata Digitalne elektronike

<c:[SRFlipFlops], rdf:[type], fer1:[Concept]>
<c:[SRFlipFlops], fer1:[conceptLabel], "T Bistabili">
<c:[SRFlipFlops], fer1:[extendsConcept], c:[FlipFlops]>

<d:[SRFlipFlops.html], rdf:[type], fer1:[WebPage]>
<d:[SRFlipFlops.html], fer1:[isAbout], c:[SRFlipFlops]>

<c:[Registers], rdf:[type], fer1:[Concept]>
<c:[Registers], fer1:[conceptLabel], "Registri">
<c:[Registers], fer1:[dependsOn], c:[FlipFlops]>

<d:[Registers.html], rdf:[type], fer1:[WebPage]>
<d:[Registers.html], fer1:[isAbout], c:[Registers]>

<c:[ShiftRegisters], rdf:[type], fer1:[Concept]>
<c:[ShiftRegisters], fer1:[conceptLabel], "Posmačni registri">
<c:[ShiftRegisters], fer1:[extendsConcept], c:[Registers]>

<d:[ShiftRegisters.html], rdf:[type], fer1:[WebPage]>
<d:[ShiftRegisters.html], fer1:[isAbout], c:[ShiftRegisters]>

<c:[Counters], rdf:[type], fer1:[Concept]>
<c:[Counters], fer1:[conceptLabel], "Brojila">
<c:[Counters], fer1:[dependsOn], c:[FlipFlops]>

<d:[Counters.html], rdf:[type], fer1:[WebPage]>
<d:[Counters.html], fer1:[isAbout], c:[Counters]>

<c:[SynchronousCounters], rdf:[type], fer1:[Concept]>
<c:[SynchronousCounters], fer1:[conceptLabel], "Sinkrona brojila">
<c:[SynchronousCounters], fer1:[extendsConcept], c:[Counters]>

<d:[SynchronousCounters.html], rdf:[type], fer1:[WebPage]>
<d:[SynchronousCounters.html], fer1:[isAbout], c:[SynchronousCounters]>

<c:[AsynchronousCounters], rdf:[type], fer1:[Concept]>
<c:[AsynchronousCounters], fer1:[conceptLabel], "Asinkrona brojila">
<c:[AsynchronousCounters], fer1:[extendsConcept], c:[Counters]>

<d:[AsynchronousCounters.html], rdf:[type], fer1:[WebPage]>
<d:[AsynchronousCounters.html], fer1:[isAbout], c:[AsynchronousCounters]>

Slika 6-1. Nastavak

Slika 6-1 prikazuje primjer u kojem navodi da postoje koncepti poput bistabila, registara i sl. Za te koncepte postoji materijal iz kojega se o njima može naučiti u obliku odgovarajućih Web stranica. Opisane su i međusobne ovisnosti koncepata, pa je tako jasno da je za razumijevanje brojila potrebno prethodno razumjeti bistabil, koji predstavlja temeljnu građevnu komponentu brojila.

Zgodno je spomenuti da se uz ovakav opis kolegija može razmišljati i o sustavima koji će ponuditi učeniku mogućnost učenja odabranih koncepata, nudeći pri tome materijale o pravim konceptima u pravo vrijeme tj. pravim redoslijedom (što je u klasičnom obliku nastave uloga predavača). Također, umjesto veza koncepata s

odgovarajućim HTML dokumentima veze se mogu uspostaviti i s "aktivnim" sadržajima učenja, odnosno SCO-ovima (iz SCORM-a).

Jednom kada imamo kolegij formaliziran na opisani način, može se krenuti na proširenje definirane ontologije u svrhu potpore inteligentnom vođenju e-ispitivanja.

6.2 Proširenje ontologije feronto1

Model StudTest nije model sustava e-učenja, i prethodno definirani opis koncepata kolegija i izrada prikladnih materijala za učenja ne spada u njegov djelokrug. Međutim, za potrebe inteligentnog vođenja e-učenja pretpostavit ćemo da takav opis postoji. Ono što je još potrebno napraviti jest uvesti proširenje korištene ontologije novim konceptima i relacijama koje će uzeti u obzir vezu između koncepata koje studenti uče na nastavi i zadataka definiranih u okviru modela StudTest.

Definirajmo potrebna proširenja. Od koncepata, uvodimo jedan novi koncept *Problem*. Taj koncept predstavlja jedan konkretan zadatak. Od relacija također uvodimo jednu novu relaciju: *asksAbout*, koja povezuje neki zadatak s nekim konceptom. Ovo je sumirano tabličnim prikazom (Tablica 6-4, Tablica 6-5).

Tablica 6-4. Proširenje ontologije feronto1 novim konceptima

Naziv koncepta	Opis
Problem	Zadatak koji provjerava znanje studenta.

Tablica 6-5. Proširenje ontologije feronto1 novim relacijama

Naziv relacije	Opis
asksAbout	Definira provjeravanje znanja nekog pojma nekim zadatkom. Primjer: <c:[BooleanAlgebra], rdf:[type], fer1:[Concept]> <c:[BooleanAlgebra], fer1:[conceptLabel], "Booleova algebra"> <z:[1], rdf:[type], fer2:[Problem]> <z:[1], fer2:[asksAbout], c:[BooleanAlgebra]> <z:[2], rdf:[type], fer2:[Problem]> <z:[2], fer2:[asksAbout], c:[BooleanAlgebra]>

Uz ova proširenja sada je moguće dopuniti prethodni primjer. Pretpostavimo da je za svaki koncept iz primjera definiran po jedan zadatak. Nakon što se zadaci unesu u sustav, prema zahtjevu iz poglavlja 4.6, sustav će korisniku obznaniti URI-je svih zadataka. U tom trenutku korisnik može generirati dopunu opisa kolegija (Slika 6-2).

Ovim su primjerom definirani koncepti dopunjeni zadacima koji ispituju njihovo razumijevanje. Pogledamo li sada što je sve poznato o konceptu npr. c:[Registers], dolazimo do koncepata i veza koje prikazuje Slika 6-3.

Ostalo je još za pogledati kako se u modelu StudTest ovakvi opisi mogu koristiti radi inteligentnog vođenja ispitnog procesa.

<z:[1], rdf:[type], fer2:[Problem]>
<z:[1], fer2:[asksAbout], c:[FlipFlops]>

<z:[2], rdf:[type], fer2:[Problem]>
<z:[2], fer2:[asksAbout], c:[FlipFlopTransitionTable]>

<z:[3], rdf:[type], fer2:[Problem]>
<z:[3], fer2:[asksAbout], c:[DFlipFlops]>

<z:[4], rdf:[type], fer2:[Problem]>
<z:[4], fer2:[asksAbout], c:[TFlipFlops]>

<z:[5], rdf:[type], fer2:[Problem]>
<z:[5], fer2:[asksAbout], c:[JKFlipFlops]>

<z:[6], rdf:[type], fer2:[Problem]>
<z:[6], fer2:[asksAbout], c:[SRFlipFlops]>

<z:[7], rdf:[type], fer2:[Problem]>
<z:[7], fer2:[asksAbout], c:[Registers]>

<z:[8], rdf:[type], fer2:[Problem]>
<z:[8], fer2:[asksAbout], c:[ShiftRegisters]>

<z:[9], rdf:[type], fer2:[Problem]>
<z:[9], fer2:[asksAbout], c:[Counters]>

<z:[10], rdf:[type], fer2:[Problem]>
<z:[10], fer2:[asksAbout], c:[SynchronousCounters]>

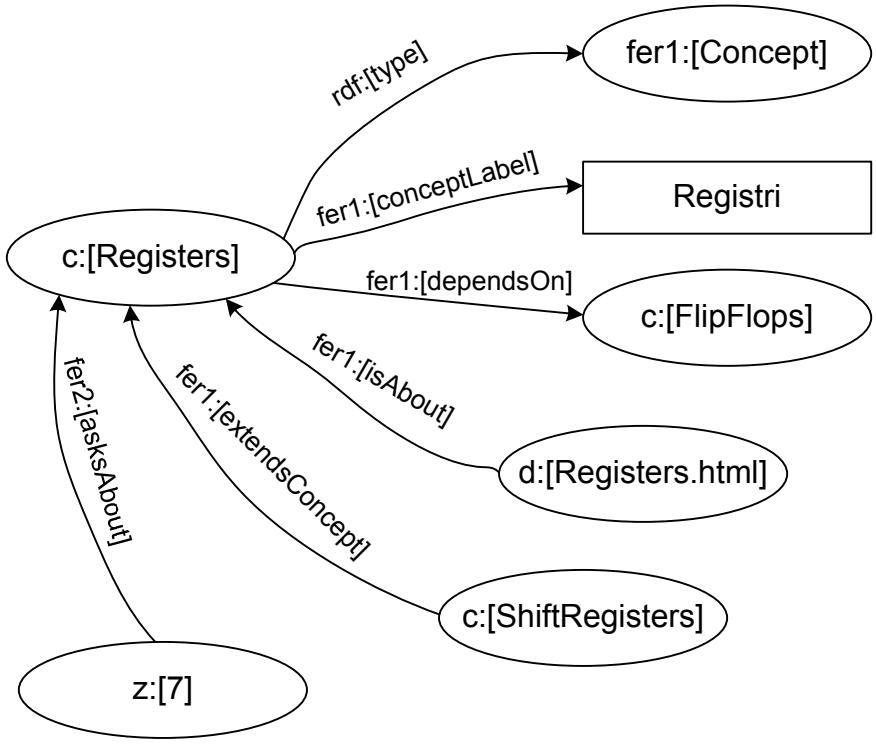
<z:[11], rdf:[type], fer2:[Problem]>
<z:[11], fer2:[asksAbout], c:[AsynchronousCounters]>

Slika 6-2. Dopuna ontološkog opisa kolegija vezama sa zadacima

6.3 Komponente za inteligentno vođenje ispitnog procesa

Za vođenje ispitnog procesa je u sustavu StudTest zadužena samo jedna komponenta: upravljač provjere znanja (TestController). Dodavanje funkcionalnosti inteligentnog vođenja ispitnog procesa stoga se svodi na implementaciju odgovarajućeg upravljača koji će u sebi sadržati potrebne algoritme umjetne inteligencije. Pogledajmo jedan primjer.

Neka je definiran upravljač provjere znanja AITestController. Koje parametre ovaj upravljač nudi na uređivanje po uključenju u opisnik provjere znanja? To dakako ovisi o odabranom algoritmu umjetne inteligencije. Primjerice, neka nudi dva parametra na uređivanje: popis uključenih koncepata, i popis isključenih koncepata. Prvi popis definira sve koncepte od kojih smije započeti ispitivanje, no ne definira donju granicu (primjerice, ako u popis dodamo koncept c:[ShiftRegisters], upravljač bi slijeđenjem odgovarajućih veza poput fer1:[extendsConcept] mogao doći do koncepta c:[Registers] (i na sličan način do ostalih koncepata). Osiguranje da ispitivanje neće odmaći predaleko od zadanih koncepata tada se može postići popisom isključenih koncepata koji će prekinuti veze po kojima se skup koncepata širi. Za potrebe primjera pretpostavimo da je u popisu uključenih koncepata samo c:[ShiftRegisters].



Slika 6-3. Sve poznato o c:[Registers]

Nakon definiranja opisnika provjere znanja, pretpostavimo da prvi student zatraži svoj primjerak provjere znanja. U tom trenutku od upravljača AITestController poziva se metoda `prepareTestInstance()`. Ova metoda sada može pogledati do kojih sve koncepata može doći krenuvši od popisa uključenih koncepata i izostavljanjem koncepata s popisa isključenih koncepata (pretraživanjem ontološkog opisa). Da bismo mogli odgovoriti na ovo pitanje, treba još definirati i koje se veze smiju slijediti (neka to budu veze fer1:[dependsOn] te fer1:[extendsConcept]). Označimo skup svih dosežljivih koncepata sa S. Za sve koncepte iz skupa S upravljač na temelju ontološkog opisa dolazi do skupa Z svih zadataka koji su vezani uz te koncepte. Odabire se jedan zadatak iz skupa Z za prikaz studentu. Kao moguće akcije studentu se nudi samo "Sljedeći" i "Završi test".

Zadatak se prikazuje studentu, student odgovara na pitanje i odabire akciju "Sljedeći".

Nad upravljačem AITestController poziva se metoda `onAction()`. U toj metodi upravljač postavljeni zadatak šalje na vrednovanje i čeka. Čim završi proces vrednovanja, upravljač analizira mjeru točnosti zadatka. Ukoliko je zadatak točan, kao sljedeći se može odabrati neki od zadataka koji primjerice ispituje koncepte koji nisu temeljniji od trenutno ispitanog koncepta (ako je student točno odgovorio na pitanje o posmačnom registru, tada se očekuje da razumije što je to registar – što je definirano vezom fer1:[extendsConcept]). Ukoliko studentov odgovor nije točan, moguće je više smjerova kojima se može krenuti. Jedan je smjer odluka da student očito ne razumije taj koncept, pa ga se nastavlja pitati nešto sasvim nevezano s tim konceptom (uz odgovarajuće smanjenje konačne ocjene). Druga je mogućnost utvrditi što to točno student ne razumije, pa krenuti u ispitivanje onih koncepata kojih je ispitani koncept proširenje, ili koje treba razumjeti da bi se razumio ispitani

koncept. Nakon što je novi zadatak odabran, stvara se primjerak tog zadatka i nudi studentu na rješavanje, čime se upravo opisani proces ponavlja.

Također, se može pretpostaviti da će stvarne implementacije upravljača AITestController odjednom odabirati više zadataka, a ne jedan po jedan, jer proces odabira zadataka, odnosno utvrđivanja što su potencijalni kandidati, može biti vrlo zahtjevan (u smislu posla koji je potrebno obaviti kako bi se pripremile odgovarajuće strukture podataka, tako da se jednom kada se ovo obavi više isplati odmah odabrati veći broj zadataka).

Ovdje prikazan primjer inteligentnog vođenja ispitnog procesa predstavlja osnovu na koju će se u budućnosti usmjeriti daljnja istraživanja, i prototipne implementacije.

7 Implementacija

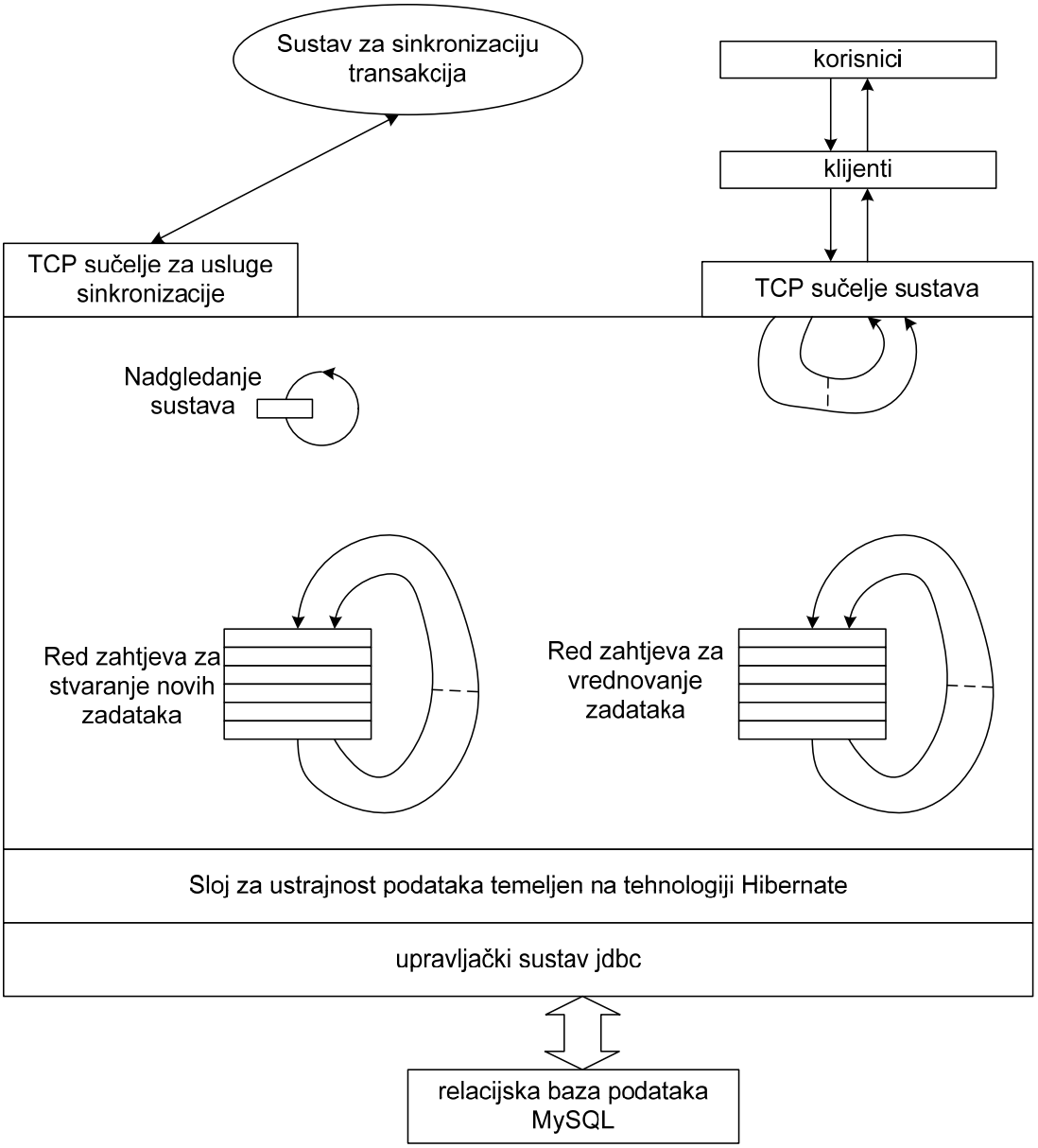
Prototipna implementacija modela StudTest u protekle je dvije godine razrađena i korištena na dva kolegija sveučilišne razine: Digitalnoj elektronici (po programu FER1), te Digitalnoj logici (po novom "Bolonjskom" programu FER2), što je omogućilo realno sagledavanje mogućnosti modela, kao i utvrđivanje performansi sustava. U načelu, može se konstatirati da sustav pokazuje zadovoljavajuće performanse.

Za implementaciju prototipne verzije sustava odabran je programski jezik Java. Razloga za to ima puno, a nekoliko najvažnijih je nabrojano u nastavku:

- Java je objektno orijentirani programski jezik, čime je osigurano prirodno preslikavanje koncepata modela u odgovarajuće razrede objektno orijentirane paradigme,
- Java je platformski nezavisna, što znači da jednom napisan sustav može raditi bez ikakvih promjena na poslužiteljima koje pogone različiti operacijski sustavi (platforme MS Windows, platforme Unix/Linux, itd.),
- za izradu zadataka koji za potrebe rješavanja nude bogato grafičko korisničko sučelje nužno je osmisliti takvu tehnologiju koja će postavljene zahtjeve moći ispuniti na bilo kojem popularnijem operacijskom sustavu na kojem radi rješavatelj zadataka; jedina danas raspoloživa tehnologija s tim karakteristikama su Java appleti [37]; kada bismo se ograničili samo na potporu korisnicima s platforme Windows, tada bi to mogao biti i ActiveX; to je, međutim, u suprotnosti s idejom o podršci širokom spektru podržanih platformi,
- kako je odabrana tehnologija prikaza složenih zadataka "Java Applets", prirodno je da se i na poslužiteljskoj strani koristi tehnologija Java, koja jedina može bez problema direktno komunicirati s appletima,
- Java podržava izuzetno lagano korištenje mrežnih tehnologija, koje čine okosnicu prototipne implementacije modela.

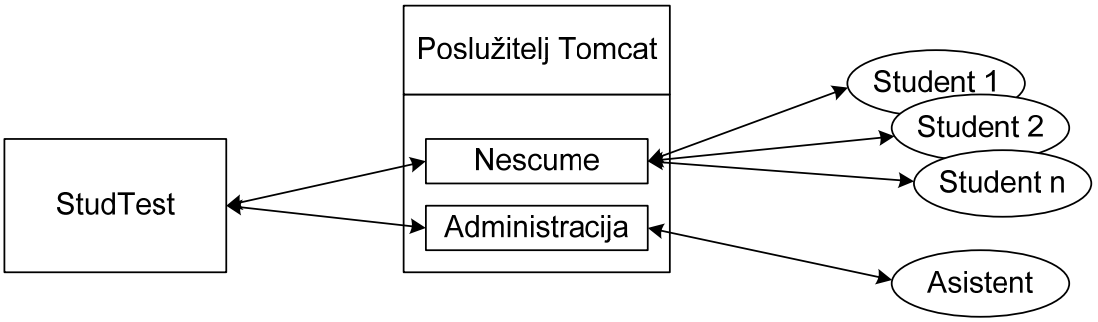
Zahvaljujući iskustvima stečenim sa sustavom WODLS, prototipna implementacija modela StudTest je projektirana tako da riješi najveće probleme koje je imao navedeni sustav. Svakako je najveći problem bila sporost prilikom predaje provjera znanja i odbijanje ocjenjivanja rezultata provjera koje su s klijentske strane bile predane na vrijeme ali zbog opterećenja sustava nisu bile prihvaćene na vrijeme na poslužiteljskoj strani. Sve vremenski zahtjevne operacije unutar prototipne implementacije su stoga izvedene kao asinkrone.

Slika 7-1 prikazuje osnovnu struktura samog sustava. Gledano izvana, sustav je crna kutija koja nema svoje korisničko sučelje, već se sva interakcija s okolinom obavlja isključivo kroz TCP sučelje. Pri tome korisnik ne može direktno komunicirati sa sustavnom. Naime, namjena sustava je ostvarivanje podsustava za ispitivanje, a sam se podsustav za pružanje korisničkog sučelja oslanja na potporu sustava e-učenja koji podsustav e-ispitivanja koristi.



Slika 7-1. Građa prototipne implementacije sustava StudTest

Ulogu sustava e-učenja u prototipnoj implementaciji ima sustav Nescume, kroz koji studenti pristupaju uslugama e-ispitivanja (Slika 7-2).



Slika 7-2. Uporaba sustava StudTest kroz sustav e-učenja

Za potrebe pohrane svih podataka sustav koristi relacijsku bazu podataka, jer je to danas provjerena i pouzdana tehnologija, s nizom izvrsnih implementacija. Za potrebe ovog sustava odabrana je implementacije otvorenog koda (engl. Open Source) MySQL [39], koja je dostupna i za platforme Windows i Unix/Linux. Pri tome sustav direktno ne koristi usluge ove baze, već s njom komunicira kroz podsustav za preslikavanje između objekata i relacija (termin poznat kao O-R preslikavanje, engl. O-R mapping) Hibernate [38].

Za potrebe sinkronizacije podatkovnih transakcija (prilikom rada s bazom podataka, odnosno prilikom manipulacija s objektima) koristi se izdvojeni podsustav za sinkronizaciju transakcija čije je sučelje također izvedeno uporabom TCP protokolnog sloga.

Dvije najzahtjevnije vrste zadatka u sustavu su zadaci stvaranja novih primjeraka zadataka, te zadaci vrednovanja studentskih rješenja. Kako bi se osigurale prihvatljive performanse i funkcioniranje sustava kao cjeline, ova dva zadatka riješena su asinkrono. Pri tome u sustavu postoje dva prioritetna reda: red zahtjeva za stvaranje novih zadataka, u koji se umeću zahtjevi za stvaranjem primjeraka zadataka, te red zahtjeva za vrednovanje zadataka. Ova dva reda poslužuje podesiv broj dretvi radnika (engl. worker threads), čime je osigurano da sustav neće u slučaju istovremene predaje velikog broja provjera znanja krenuti u istovremeno vrednovanje svih provjera (i doživjeti zagušenje), već će ovaj posao obavljati malo po malo. Redovi su prioritetni, kako bi se osigurala potpora prilagodljivom vođenju ispitnog postupka, gdje se zadaci moraju stvarati i vrednovati u najkraćem moguće roku. Ovi se redovi mogu izvesti i kao raspodijeljeni, čime se osigurava raspodijeljen rad više implementacija modela StudTest u svrhu povećanja performansi.

7.1 Implementacije koncepata modela StudTest

Kako model StudTest ne propisuje koje je sve konkretne implementacije određenih koncepata potrebno napraviti, u nastavku će biti opisane implementacije komponenti koje su korištene za potrebe kolegija Digitalna elektronika i Digitalna logika.

7.1.1 Upravljači provjere znanja

U ovom poglavlju dan je kratak pregled trenutno implementiranih upravljača provjera znanja. Trenutno je implementiran samo jedan upravljač, koji se je pokazao dovoljnim za sve potrebe kolegija Digitalna elektronika i Digitalna logika.

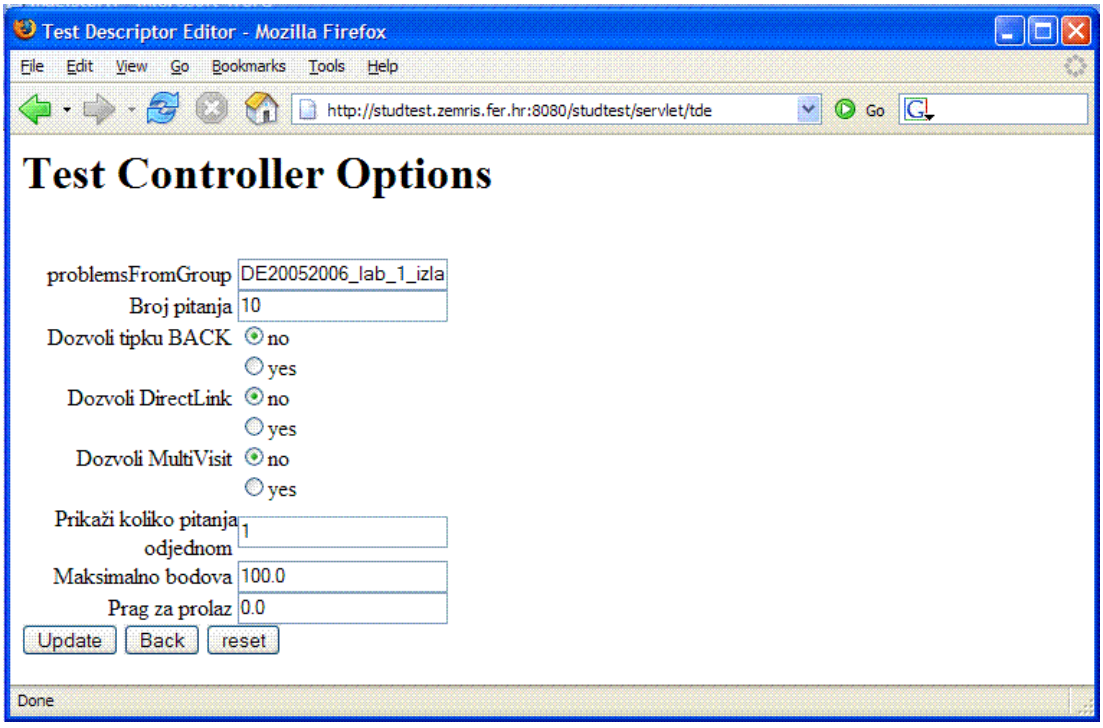
7.1.1.1 SimpleTestController

Ovaj upravljač provjere znanja definira razred testova fiksnog broja zadataka, pri čemu se zadaci biraju iz ponuđene grupe zadataka korištenjem ugrađene potpore inteligentnim postupcima (sučelje IReasoner, poglavlje 5). Test se definira kao slijed od k n-torki pitanja (čime je ukupni broj pitanja jednak k*n), pri čemu se sva pitanja odabiru u trenutku stvaranja testa. Obzirom na ovako veliku ograničenost oblika testa, upravljač nudi čitav niz opcija kojima se detaljnije određuje što sve student može koristiti tijekom pisanja testa. Opcije s kratkim opisom navedene su u nastavku.

Tablica 7-1. Parametri koje nudi komponenta SimpleTestController

Naziv parametra	Opis parametra
Grupa	odabir grupe iz koje se biraju zadaci koji ulaze u test
Broj pitanja	koliko pitanja treba odabrati za test
Dozvoli tipku BACK	smije li se tijekom pisanja testa koristiti tipka BACK
Dozvoli DirectLink	je li tijekom pisanja testa dozvoljeno koristiti direktnu kretanje do svakog zadatka, ili se za kretanje smiju koristiti samo tipke BACK (koja je moguće isključena) i NEXT
Dozvoli MultiVisit	može li se pisanje testa zamrznuti, i nastaviti u nekom drugom trenutku; ova se opcija tipično koristi kod pisanja domaćih zadaća i slične vrste testova, kod koje student testu može pristupati više puta (više dana u tjednu), prije no što ga preda
Broj pitanja odjednom	koliko pitanja treba prikazati odjednom; tipično je postavljeno na 1.
Maksimalno bodova	koliko se maksimalno bodova može dobiti na ovom testu
Prag za prolaz	koliko je minimalno bodova potrebno skupiti da bi se test smatrao položenim

Slika 7-3 prikazuje administracijsko sučelje za podešavanje parametara ove komponente (stvarni primjer). Sučelje je stvoreno dinamički.



Slika 7-3. Sučelje za podešavanje parametara pisanja provjere znanja

7.1.2 Provjerivači dozvole početka pisanja provjere znanja

Model StudTest definira koncept provjerivača dozvole početka pisanja provjere znanja, no ne propisuje niti jednu implementaciju. U nastavku su opisane do sada ostvarene implementacije ovih komponenti.

7.1.2.1 QueueStartChecker

Zadaća ovog provjerivača početka pisanja provjere je omogućavanje početka pisanja kroz red za omogućavanje. Koristi se u slučaju kada asistent želi omogućiti početak pisanja svima u istom trenutku. Ukoliko je ovaj provjerivač uključen u opisnik provjere, tada se student prilikom zahtjeva za početkom pisanja provjere automatski prijavljuje na red za omogućavanje, pri čemu se dozvola za početkom pisanja odgađa.

Asistent u svakom trenutku može pogledati koje su sve osobe zatražile provjeru, te ih omogućiti svima u istom trenutku.

U okviru kolegija Digitalna logika ovaj provjerivač koristi se za omogućavanje početka pisanje izlaznog testa na laboratorijskim vježbama. Kako se laboratorijske vježbe izvode u 9 laboratorija paralelno, definirano je 9 redova za omogućavanje provjera. U svakom laboratoriju studenti se prijavljuju u svoj red za omogućavanje, čime se osigurava da svi studenti u jednom laboratoriju započnu istovremeno pisanje izlaznog testa.

7.1.2.2 TimeFrameStartChecker

Zadaća ovog provjerivača početka pisanja provjere jest omogućavanje početka pisanja provjere u zadanom vremenskom razdoblju. U okviru opisnika provjere, provjerivač definira dva parametra: početak i kraj razdoblja.

U okviru kolegija Digitalna logika ovaj provjerivač se koristi kod pisanja domaćih zadaća, koje studenti moraju riješiti u zadanom tjednu nastave. Zadatak provjerivača je onemogućavanje prijevremenog pristupa zadaći, ili pak rješavanje zadaće nakon zadanog tjedna.

7.1.2.3 TimeWindowStartChecker

Zadaća ovog provjerivača je omogućavanje početka pisanja provjere samo u zadanom vremenskom intervalu nakon što je dana dozvola za početak pisanja. Ovaj se provjerivač usko integrira s provjerivačem QueueStartChecker, te u okviru opisnika provjere definira dva parametra: naziv reda za omogućavanje provjere te trajanje intervala unutar kojeg se provjera još može započeti pisati, nakon što je dozvola dana. Ukoliko student pokuša započeti pisanje provjere nakon isteka definiranog vremenskog intervala, dozvola za početak pisanja se automatski povlači i pristup provjeri se onemogućuje.

U okviru kolegija Digitalna logika ovaj provjerivač koristi se za omogućavanje početka pisanje izlaznog testa na laboratorijskim vježbama, pri čemu se interval tipično definira kao 1 minuta. Ovime se postiže istovremeni početak pisanja svih studenata.

7.1.2.4 IPAddressStartChecker

Zadaća ovog provjerivača početka pisanja provjere jest omogućavanje početka pisanja provjere samo sa zadanih IP adresa. U okviru opisnika provjere, provjerivač definira jedan parametar: skup raspona dozvoljenih IP adresa.

U okviru kolegija Digitalna logika ovaj provjerivač koristi se za omogućavanje početka pisanje izlaznog testa na laboratorijskim vježbama, kako bi se osiguralo da su svi studenti koji su zatražili test doista u laboratoriju pod nadzorom asistenata.

7.1.2.5 PasswordProtectionStartChecker

Zadaća ovog provjerivača početka pisanja provjere jest omogućavanje početka pisanja provjere samo studentima koji poznaju zaporku potrebnu za njeno omogućavanje. Ovaj provjerivač definira alternativni mehanizam provjere da se student koji piše provjeru doista nalazi pod nadzorom asistenta (prva mogućnost je kontrola IP adresa).

7.1.2.6 PassedTestPrerequisiteStartChecker

Zadaća ovog provjerivača početka pisanja provjere jest omogućavanje početka pisanja samo studentima koji su položili sve provjere koje su preduvjet za aktualnu provjeru.

U okviru kolegija Digitalna elektronika ovaj provjerivač koristi se za omogućavanje početka pisanje izlaznog testa na laboratorijskim vježbama, samo onim studentima koji su položili ulazni test, čime je spriječeno da asistent nehotice omogući pisanje izlaznog testa studentu koji nije položio ulazni test (a iz nekog je razloga ostao u laboratoriju, i na kraju zatražio izlazni test).

7.1.3 Nadglednici tijeka pisanja provjere znanja

Za potrebe oba kolegija se je do sada pokazala potreba za samo jednim nadglednikom tijeka pisanja provjere znanja, koja je opisana u nastavku.

7.1.3.1 TestDurationSupervisor

Zadaća ovog nadglednika je nadzor vremenskog ograničenja vezanog uz pisanje same provjere. Nadglednik definira tri parametra: vrijeme pisanja provjere, vrijeme kašnjenja predaje provjere te do kada provjera mora završiti. Sva tri parametra ne moraju biti popunjena. Vrijeme pisanja provjere definira koliko vremena student ima na raspolaganju za njeno rješavanje, mjereno od trenutka početka pisanja. Vrijeme kašnjenja predaje definira koliko vremena smije proteći nakon što istekne vrijeme pisanja provjere, unutar kojeg će se ona još prihvatiti (i zatim poslati na ocjenjivanje). Parametar do kada pisanje provjere mora završiti definira apsolutni vremenski trenutak (datum + vrijeme) kada pisanje najkasnije mora završiti.

U okviru kolegija Digitalna logika ovaj se provjerivač koristi na dva načina. Kod izlaznih testova na laboratorijskim vježbama definiraju se prva dva parametra (vrijeme pisanja testa na 15 minuta, te kašnjenje predaje na 1 minutu). Kod domaćih zadaća koristi se zadnji parametar koji definira apsolutno ograničenje kada domaća zadaća mora završiti.

7.1.4 Ocjenjivači

U ovom poglavlju dan je kratak pregled do sada implementiranih ocjenjivača provjera. Trenutno je implementiran samo jedan ocjenjivač, koji se pokazao dovoljnim za sve potrebe kolegija Digitalna elektronika i Digitalna logika.

7.1.4.1 SimpleGrader

Ovaj ocjenjivač postupak ocjenjivanja zadatka provodi u skladu s programom definiranim u opisniku testa. Nakon što ocjenjivači primjerka zadatka (koji su dio generatora problema) završe posao, svakom je primjerku zadatka pridružena mjera točnosti, što je broj u intervalu od 0 do 1. Prilikom pisanja testa, student za svaki zadatak može definirati mjeru pouzdanosti, koliko smatra da je odgovor koji je odabrao točan (ili ako ovo ne odabire, podrazumijeva se vrijednost 1). Prilikom rada, ocjenjivač primjerka zadatka osim mjere točnosti zadatku pridružuje još i Booleovu vrijednost isSolved. Na temelju ova tri parametra ocjenjivač zadatka može ostvariti čitav niz načina bodovanja, i skladu s programom koji mu se definira.

Slika 7-4 prikazuje primjer programa za ocjenjivanje koji u obzir uzima samo mjeru točnosti.

```
if $isCorrect then
  return 10;
else
  return -5;
end if;
```

Slika 7-4. Primjer ocjenjivača s negativnim bodovima za netočne i neriješene zadatke

Iskustvo pokazuje da ovaj način ocjenjivanje kod studenata izaziva veliko nezadovoljstvo – naime, zadaci koji nisu rješavani smatraju se netočnim. Ukoliko netočan odgovor nosi negativne bodove – ocjenjivanje se doima nepravednim.

Slika 7-5 prikazuje primjer programa koji u obzir uzima i parametar je li zadatak rješavan ili nije.

```
if $isSolved then
  if $isCorrect then
    return 10;
  else
    return -5;
  end if;
else
  return 0;
end if;
```

Slika 7-5. Primjer ocjenjivača koji negativne bodove daje samo za netočne odgovore

Prethodno prikazani načini ocjenjivanja kod složenih zadataka koji mogu biti djelomično točni također nisu baš najprimjereniji. Naime, poslužitelj StudTest osmišljen je s namjerom podrške izradi složenih zadataka kod kojih će ocjenjivač primjerka zadatka generirati mjeru točnosti u kojoj će se naći i brojevi iz unutrašnjosti intervala [0,1], a ne samo granice. Tada se kao “pošten” način

ocjenjivanja doživljava proporcionalna dodjela bodova mjeri točnosti rješenja. Slika 7-6 prikazuje program koji ostvaruje ovakvo ocjenjivanje.

```
if $isSolved then
  return proportional 10;
else
  return 0;
end if;
```

Slika 7-6. Ocjenjivač koji daje broj bodova proporcionalan mjeri točnosti

Ukoliko se u obzir želi uzeti i mišljenje studenta definirano mjerom njegove pouzdanosti u točnost rješenja, može se koristiti program oblika sličnog prethodnom primjeru (Slika 7-7).

```
if $isSolved then
  return proportional confidential 10;
else
  return 0;
end if;
```

Slika 7-7. Ocjenjivač s brojem bodova proporcionalnim mjerama točnosti i pouzdanosti

Ako je pak potrebno različite zadatke različito bodovati, može se iskoristiti program koji u obzir uzima identifikator zadatka (koji je sastavljaču pitanja vidljiv u trenutku izrade/uređivanja pitanja). Slika 7-8 prikazuje takav slučaj.

```
if $isSolved then
  if $problemID IS 12 then
    return proportional confidential 30;
  else
    return proportional confidential 5;
  end if;
else
  return 0;
end if;
```

Slika 7-8. Ocjenjivač koji daje različit broj bodova ovisno o ocjenjivanom zadatku

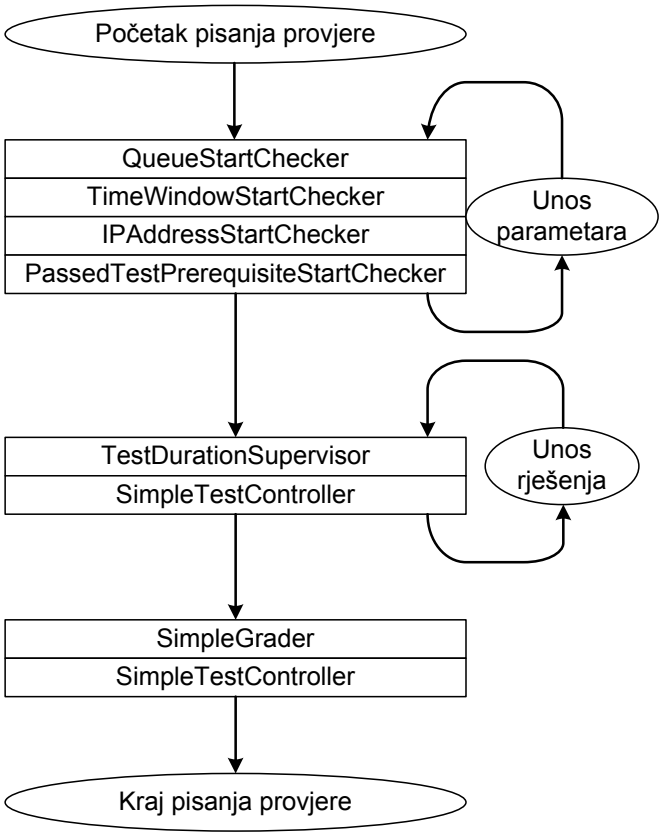
U opisanom slučaju pretpostavka je da je zadatak s identifikatorom 12 težak i vremenski zahtjevan, te mu se daje punih 30 bodova (ako se riješi apsolutno točno), dok svi ostali zadaci nose najviše 5 bodova.

Kompleksniji načini ocjenjivanja, tj. uvažavanje težine pojedinih zadataka također je podržano, ali zbog ograničenosti rada nije prikazano.

7.1.5 Uporaba na kolegijima

U ovom odjeljku prikazan je način uporabe prethodno opisanih komponenti na primjerima kolegija Digitalna elektronika i Digitalna logika.

Na kolegiju Digitalna elektronika sustav se koristi za pisanje ulaznih i izlaznih testova na laboratorijskim vježbama. Ulazni test piše se na početku svake vježbe kako bi se osiguralo da su svi studenti kod kuće pročitali upute za laboratorijske vježbe, i napravili odgovarajuću pripremu. Ovaj test je eliminacijskog karaktera, što



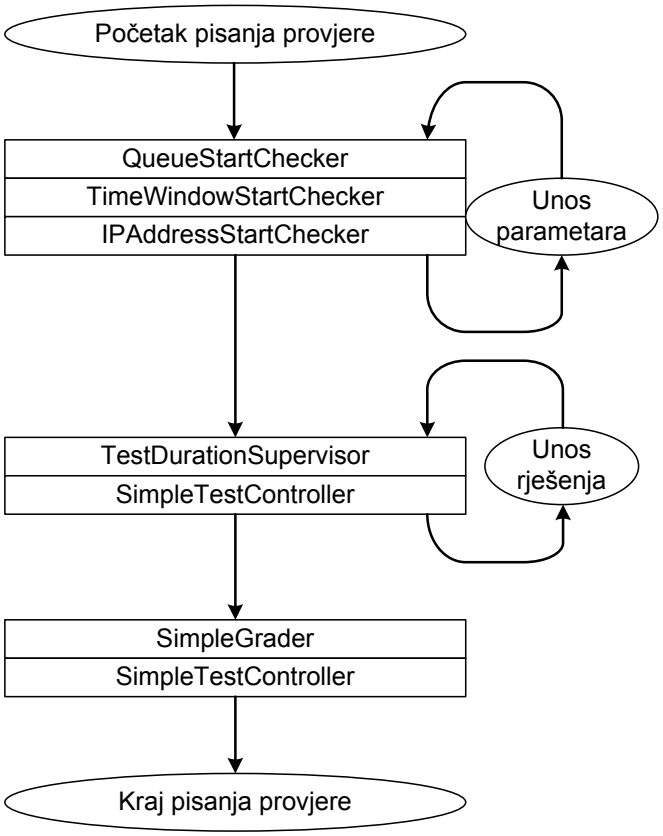
Slika 7-9. Pisanje ulaznog testa na kolegiju Digitalna elektronika

znači da se u opisniku provjere znanja definira minimalni broj bodova (prag) koji student mora sakupiti, kako bi mogao ostati u laboratoriju i započeti izvođenje same vježbe. U skladu s politikom kolegija, svim se studentima provjera omogućava istovremeno, kako bi se smanjila mogućnost prepisivanja. Za rješavanje 10 zadataka koliko ih je definirano za provjeru, student ima 10 minuta, pri čemu se prikazuje zadatak po zadatak, i tipka BACK je onemogućena.

Prilikom odabira komponenti koje ulaze u opisnik provjere znanja, stoga se odabiru provjerivači koji će osigurati da svi doista započnu istovremeno, te da se provjeri pristupa samo iz nadziranog laboratorija. Slika 7-9 prikazuje postupak pisanja ulaznog testa.

Na kraju svake laboratorijske vježbe studenti koji su polaganjem ulaznog testa stekli pravo na rad u laboratoriju pišu izlazni test. Svrha ovog testa jest provjeriti rezultate rada u laboratoriju, kao i odgovarajuće poznavanje teorijskih osnova na kojima se zasniva sama vježba. Zbog toga bodovi s izlaznog testa direktno doprinose ukupnom broju bodova koje je student može ostvariti na kolegiju, i time utječu na ocjenu.

Iz navedenih razloga, kod izlaznog testa se u opisnik provjere znanja uključuje još i provjerivač PassedTestPrerequisiteStartChecker, koji će prije davanja dozvole pristupa izlaznom testu provjeriti je li student doista položio ulazni test (i time rasteretiti asistenta ovog posla). Ukoliko student nije položio ulazni test, a ipak je nekako uspio ostati u laboratoriju, pristup izlaznom testu bit će mu uskraćen. Slika 7-10 prikazuje postupak pisanja izlaznog testa.

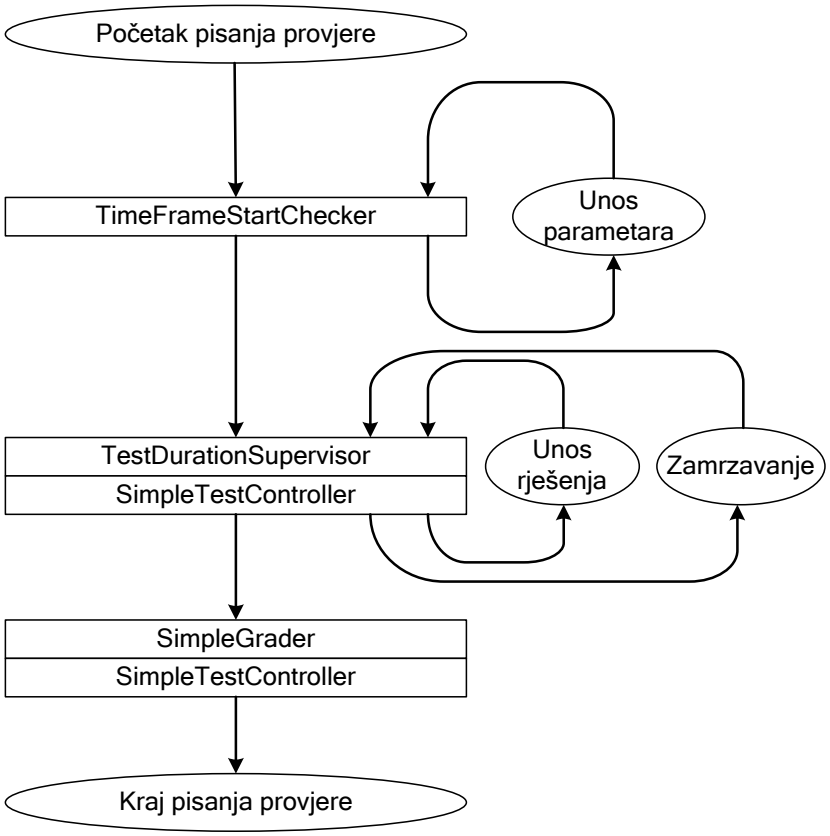


Slika 7-10. Pisanje izlaznog testa na kolegiju Digitalna elektronika

Za razliku od ulaznih i izlaznih testova na laboratorijskim vježbama kolegija Digitalna elektronika, Bolonjska organizacija nastavnog procesa donijela je novosti koje u izvornoj implementaciji koncepata sustava (zapravo niti u samom modelu) nisu bile podržane. Naime, Bolonjska organizacija nastave, osim laboratorijskih vježbi gdje se znanje provjerava samo izlaznim testovima donijela je i domaće zadaće, koje se sa stanovišta sustava ponašaju gotovo kao provjere znanja, ali ne u potpunosti.

Naime, klasična provjera znanja oduvijek je povlačila pridruživanje odgovarajućeg stupnja sigurnosnih mjera: nadzor pristupnika, rješavanje uz uska vremenska ograničenja i sl. Domaće zadaće radikalno mijenjaju ovakav pogled na provjeru, jer studenti zadaći mogu pristupati višekratno, riješavati dio po dio zadataka, bez ikakvog nadzora i uz međusobnu suradnju.

Kako bi se podržao ovakav pogled na provjeru znanja, model sustava trebalo je revidirati. Zahvaljujući dobrom odvajanju funkcionalnosti pojedinih komponenti modela (engl. function separation), u modelu je trebalo revidirati samo komponentu zaduženu za vođenje ispita: upravljač. Koncept upravljač proširen je tako da kao sljedeću akciju uz standardne akcije "nastavi" i "završi" podržava i akciju "zamrzni" (engl. suspend) kojom se sam postupak pisanja provjere zamrzava na određeno vrijeme. Dakako, revidirana je i implementacija odgovarajuće komponente (u ovom slučaju komponente SimpleTestController). Slika 7-11 prikazuje izvođenje domaće zadaće.



Slika 7-11. Pisanje domaće zadaće na kolegiju Digitalna logika

7.1.6 Implementirani tipovi zadataka

Tip zadatka predstavlja način na koji ispitanik doživljava zadatak odnosno način na koji se na pitanje odgovara (da li odabirom odgovora, upisivanjem broja, povezivanjem parova i sl.). Također, važno je napomenuti da tip zadatka ni na koji način ne određuje hoće li zadatak biti dinamičan ili ne. Primjerice, ako zadatak prilikom stvaranja dinamički utvrđuje parametre na temelju kojih će se korisniku iscrtati slika, zadatak ćemo zvati dinamičkim. Ako sastavljač provjere tekst zadatka definira unaprijed, zadatak ćemo zvati statičkim. No, u oba slučaja, zadatak korisniku možemo predstaviti na isti način: kao ABC pitalicu s jednim točnim odgovorom. Tijekom rada na kolegijima Digitalna elektronika i Digitalna logika koji se predaju na Fakultetu elektrotehnike i računarstva, identificirana je potreba za 5 tipova zadataka, koji su potom i implementirani. Implementacije tih tipova opisane su u nastavku. Svi ovi tipovi u potpunosti podržavaju izradu statičkih te dinamičkih podvrsta.

7.1.6.1 Tip ABCSingleCorrect

Ovaj tip podržava izradu zadataka koji na odabir ispitaniku nude niz opcija, od kojih samo jedna može biti točna. U tehnologiji HTML ovakav će se tip uobičajeno prikazivati uporabom kontrole "radio-button". Za potrebe rada s ovim tipom zadatka implementirano je sučelje IAbcSingleCorrectType (Tablica 7-2), a za potrebe programskog stvaranja ovakvih zadataka proširenje tog sučelja dodatnim metodama, u obliku sučelja IAbcSingleCorrectTypeEditor (Tablica 7-3).

Tablica 7-2. Sučelje IAbcSingleCorrectType

`String[] getOptions();` Metoda za dohvat tekstova svih opcija.
`int getCorrectOptionIndex();` Metoda za dohvat pozicije točne opcije.
`int getSelectedOptionIndex();` Metoda za dohvat pozicije opcije koju je ispitanik označio.
`String getText();` Metoda za dohvat teksta zadatka.
`void setSelectedOptionIndex(int selectedOptionIndex);` Metoda za postavljanje pozicije opcije koju je ispitanik označio.
`void setExplanation(String explanation);` Metoda za postavljanje objašnjenja.

Tablica 7-3. Sučelje IAbcSingleCorrectTypeEditor

`void setOptions(String[] options);` Metoda za postavljanje tekstova svih opcija.
`void setCorrectOptionIndex(int correctOptionIndex);` Metoda za postavljanje pozicije točne opcije.
`void setText(String text);` Metoda za postavljanje teksta zadatka.
`void setHelp(String helpText);` Metoda za postavljanje pomoći u sam primjerak zadatka.
`void setProblemHelp(String helpText);` Metoda za postavljanje pomoći u zadatak.

7.1.6.2 Tip IAbcMultiCorrect

Ovaj tip podržava izradu zadataka koji na odabir ispitaniku nude niz opcija, od kojih više može biti točno. U tehnologiji HTML ovakav će se tip uobičajeno prikazivati uporabom kontrole "check-box". Za potrebe rada s ovim tipom zadatka implementirano je sučelje IAbcMultiCorrectType (Tablica 7-4), a za potrebe programskog stvaranja ovakvih zadataka proširenje tog sučelja dodatnim metodama, u obliku sučelja IAbcMultiCorrectTypeEditor (Tablica 7-5).

Tablica 7-4. Sučelje IAbcMultiCorrectType

`int getOptionsCount();` Metoda za dohvat broja opcija koje treba ponuditi ispitaniku.
`String[] getOptions();` Metoda za dohvat tekstova svih opcija.
`boolean[] getCorrectnessStatus();` Metoda za dohvat zastavica koje govore koja je opcija točna a koja nije.
`boolean[] getSelectionStatus();` Metoda za dohvat zastavica koje govore koje je opcije ispitanik označio, a koje nije.
`String getText();` Metoda za dohvat teksta zadatka.
`void setSelectionStatus(boolean[] selectionStatus);` Metoda za postavljanje zastavica koje govore koje je opcije ispitanik označio, a koje nije.
`void setExplanation(String explanation);` Metoda za postavljanje objašnjenja.

Tablica 7-5. Sučelje IAbcMultiCorrectTypeEditor

`void setOptions(String[] options);` Metoda za postavljanje tekstova svih opcija.
`void setCorrectOptions(boolean[] correctOptions);` Metoda za postavljanje zastavica koje govore koja je opcija točna a koja nije.
`void setText(String text);` Metoda za postavljanje teksta zadatka.
`void setHelp(String helpText);` Metoda za postavljanje pomoći u sam primjerak zadatka.
`void setProblemHelp(String helpText);` Metoda za postavljanje pomoći u zadatak.

7.1.6.3 Tip Text

Ovaj tip podržava izradu zadataka koji kao odgovor očekuju unos teksta, pri čemu to može biti unos jedne linije teksta, ili pak unos većeg teksta (kroz više linija). U tehnologiji HTML ovakav će se tip uobičajeno prikazivati uporabom kontrole `input` (uz type="text", za jednolinijske unose) ili pak uporabom kontrole `textarea` (za višelinijske unose). Za potrebe rada s ovim tipom zadatka implementirano je sučelje ITextType (Tablica 7-6), a za potrebe programskog

stvaranja ovakvih zadataka proširenje tog sučelja dodatnim metodama, u obliku sučelja ITextTypeEditor (Tablica 7-7).

Tablica 7-6. Sučelje ITextType

`String getText();` Metoda za dohvat teksta zadatka.
`String getValue();` Metoda za dohvat ispitanikovog odgovora.
`void setValue(String value);` Metoda za postavljanje ispitanikovog odgovora.
`String getDisplayInfo();` Metoda za dohvat načina na koji treba prikazati kutiju za unos teksta: jedna linija, mala/srednja/velika višelinijska kutija.
`String getCorrectMatcherData();` Dohvat uzorka koji opisuje točno rješenje (u svrhu automatskog ocjenjivanja bez izrade vlastitog vrednovatelja). Može biti primjerice regularni izraz.
`String getCorrectSolution();` Dohvat točnog odgovora.
`void setExplanation(String explanation);` Postavljanje objašnjenja.
`void setCorrectSolutionAvailable(boolean isAvailable);` Postavljanje zastavice podržava li ovaj zadatak izradu točnog rješenja.
`void setCorrectSolutionUnique(boolean isUnique);` Postavljanje zastavice koja govori je li generirano rješenja jedinstveno.

Tablica 7-7. Sučelje ITextTypeEditor

`void setText(String text);` Postavljanje teksta zadatka.
`void setDisplayInfo(String displayInfo);` Metoda za postavljanje načina na koji treba prikazati kutiju za unos teksta: jedna linija, mala/srednja/velika višelinijska kutija.
`String getMatcherClass();` Metoda za dohvat naziva automatskog vrednovatelja koji će se koristiti za vrednovanje ovog zadatka (ne mora se koristiti).

Tablica 7-7. Nastavak

`void setMatcherClass(String name);` Metoda za postavljanje naziva automatskog vrednovatelja koji će se koristiti za vrednovanje ovog zadatka (ne mora se koristiti).
`String getMatcherData();` Metoda za dohvat dodatnih podataka (konfiguracije) automatskog vrednovatelja.
`void setMatcherData(String data);` Metoda za pohranu dodatnih podataka (konfiguracije) automatskog vrednovatelja.
`void setCorrectMatcherData(String data);` Postavljanje uzorka koji opisuje točno rješenje (u svrhu automatskog ocjenjivanja bez izrade vlastitog vrednovatelja). Može biti primjerice regularni izraz.
`void setHelp(String helpText);` Metoda za postavljanje pomoći u sam primjerak zadatka.
`void setProblemHelp(String helpText);` Metoda za postavljanje pomoći u zadatak.

7.1.6.4 Tip TextListType

Ovaj tip podržava izradu zadataka koji kao odgovor očekuju unos liste tekstova, pri čemu svaki odgovor može biti unos jedne linije teksta, ili pak unos većeg teksta (kroz više linija). U tehnologiji HTML ovakav će se tip uobičajeno prikazivati uporabom niza kontrola input (uz type="text", za jednolinijske unose) ili pak uporabom niza kontrola textarea (za višelinijske unose). Za potrebe rada s ovim tipom zadatka implementirano je sučelje ITextListType (Tablica 7-8), a za potrebe programskog stvaranja ovakvih zadataka proširenje tog sučelja dodatnim metodama, u obliku sučelja ITextListTypeEditor (Tablica 7-9).

Tablica 7-8. Sučelje ITextListType

`String getText();` Metoda za dohvat teksta zadatka.
`String getValue();` Metoda za dohvat ispitanikovog odgovora.
`void setValue(String value);` Metoda za postavljanje ispitanikovog odgovora.
`String getDisplayInfo(int index);` Metoda za dohvat načina na koji treba prikazati određenu kutiju za unos teksta: jedna linija, mala/srednja/velika višelinijska kutija.

Tablica 7-8. Nastavak

`String getDefaultDisplayInfo();` Metoda za dohvat načina na koji treba prikazati kutiju za unos teksta: jedna linija, mala/srednja/velika všelinijska kutija, ukoliko za tu kutiju to nije eksplicitno određeno određenom metodom.
`String getCorrectMatcherData();` Dohvat uzorka koji opisuje točno rješenje (u svrhu automatskog ocjenjivanja bez izrade vlastitog vrednovatelja). Može biti primjerice regularni izraz.
`int getNumberOfElements();` Dohvat broja odgovora koje korisnik treba upisati.
`String getElementName(int index);` Naziv (labela) koji će se ispisati uz kutiju za unos zadanog odgovora.
`String getCorrectSolution(int index);` Dohvat točnog odgovora za zadani element.
`void setExplanation(String explanation);` Postavljanje objašnjenja.
`void setCorrectSolutionAvailable(boolean isAvailable);` Postavljanje zastavice podržava li ovaj zadatak izradu točnog rješenja.
`void setCorrectSolutionUnique(boolean isUnique);` Postavljanje zastavice koja govori je li generirano rješenja jedinstveno.

Tablica 7-9. Sučelje ITextListTypeEditor

`void setText(String text);` Postavljanje teksta zadatka.
`void setDisplayInfo(int index, String displayInfo);` Metoda za postavljanje načina na koji treba prikazati odabranu kutiju za unos teksta: jedna linija, mala/srednja/velika všelinijska kutija.
`void setDefaultDisplayInfo(String displayInfo);` Metoda za postavljanje načina na koji treba prikazati kutiju za unos teksta: jedna linija, mala/srednja/velika všelinijska kutija, ukoliko to eksplicitno nije definirano.
`String getMatcherClass(int index);` Metoda za dohvat naziva automatskog vrednovatelja koji će se koristiti za vrednovanje odabranog odgovora ovog zadatka (ne mora se koristiti).
`void setMatcherClass(int index, String name);` Metoda za postavljanje naziva automatskog vrednovatelja koji će se koristiti za vrednovanje odabranog odgovora ovog zadatka (ne mora se koristiti).

Tablica 7-9. Nastavak

`String getMatcherData(int index);` Metoda za dohvat dodatnih podataka (konfiguracije) automatskog vrednovatelja.
`void setMatcherData(String data);` Metoda za pohranu dodatnih podataka (konfiguracije) automatskog vrednovatelja.
`void setCorrectMatcherData(String data);` Postavljanje uzorka koji opisuje točno rješenje (u svrhu automatskog ocjenjivanja bez izrade vlastitog vrednovatelja). Može biti primjerice regularni izraz.
`void setHelp(String helpText);` Metoda za postavljanje pomoći u sam primjerak zadatka.
`void setProblemHelp(String helpText);` Metoda za postavljanje pomoći u zadatak.

7.1.6.5 Tip CustomProblemPanel

Kao potporu zadacima koji studentu moraju ponuditi kompleksno korisničko sučelje (primjerice, mogućnost crtanja sheme digitalnog/analognog sustava) definiran je ovaj tip, koji prezentaciju zadatka klijentu obavlja kroz tehnologiju Java Applets.

Pri tome ovaj tip zadatka omogućava autorima da stvore bilo koju komponentu koja nasljeđuje javax.swing.JPanel (i koja implementira složeno grafičko korisničko sučelje s izbornicima, alatnim trakama i sl), koja će kroz Java Applet biti prenesena i prezentirana korisniku, za unos njegovog rješenja. Dva su sučelja važna za potporu ovog tipa: za osnovni rad sa zadatkom sučelje ICustomProblemPanelType (Tablica 7-10), te za stvaranje samog zadatka sučelje ICustomProblemPanelTypeEditor (Tablica 7-11).

Tablica 7-10. Sučelje ICustomProblemPanelType

`String getCustomProblemPanelClassName();` Dohvat razreda koji predstavlja element korisničkog sučelja koji se prenosi korisniku.
`int getPreferedWidth();` Dohvat željene širine prenesene komponente.
`int getPreferedHeight();` Dohvat željene visine prenesene komponente.
`Object getData();` Dohvat podataka iz repozitorija koji predstavljaju sam zadatak i trenutno pohranjeni korisnikov odgovor (koristi se u svrhu inicijalizacije elementa korisničkog sučelja).

Tablica 7-10. Nastavak

`void setData(Object data);` Pohrana podataka u repozitorij koji predstavljaju trenutno pohranjeni korisnikov odgovor.
`Object getCorrectSolutionData();` Dohvat točnog rješenja zadatka.
`void setExplanation(String explanation);` Postavljanje objašnjenja.
`void setCorrectSolutionAvailable(boolean isAvailable);` Postavljanje zastavice podržava li ovaj zadatak izradu točnog rješenja.
`void setCorrectSolutionUnique(boolean isUnique);` Postavljanje zastavice koja govori je li generirano rješenja jedinstveno.

Tablica 7-11. Sučelje ICustomProblemPanelTypeEditor

`void setHelp(String helpText);` Metoda za postavljanje pomoći u sam primjerak zadatka.
`void setProblemHelp(String helpText);` Metoda za postavljanje pomoći u zadatak.

7.2 Uporaba ontologija u postojećem sustavu

Kako bi sustav ponudio dobar temelj za razvoj što šireg spektra podržanih komponenti, potpora inteligentnim postupcima neophodan je dio sustava. U sustav StudTest stoga je uključena rudimentarna potpora pohranjivanju semantičkih mreža, čime do izražaja dolazi mogućnost korištenja ontologija.

7.2.1 Potpora radu s ontologijama

U okviru StudTest poslužitelja, umjetna inteligencija može se iskoristiti na više mjesta, no prvo i osnovno što pada na pamet jest izrada adaptivnih testova. StudTest poslužitelj ovaj posao dosta olakšava, zahvaljujući svom dizajnu, koji kompletan proces upravljanja testom izolira u posebnu komponentu, koja s ostatkom sustava komunicira kroz dobro definirana sučelja.

Kao potporu radu sa semantičkim mrežama sustav zainteresiranim komponentama nudi objekt koji implementira sučelje IReasoner (Slika 7-12).

Sučelje ističe tri temeljne metode, koje podržavaju tri temeljne operacije nad uređenim trojkama: dodavanje (`addTriplet`), brisanje (`deleteTriplet`) te pretraživanje (`queryTriplet`).

```
public interface IReasoner {
  public List listProblemsInGroup(
        String groupName,
        boolean followTransitive);
  public List listSubgroupsOfGroup(
        String groupName,
        boolean followTransitive);
  public List listParentsOfGroup(
        String groupName,
        boolean followTransitive);
  public List listProblemsExclusiveWith(
        String problemID);
  public boolean getExclusiveGroupFlag(
        String groupName);
  public void setExclusiveGroupFlag(
        String groupName);
  public void addProblemInGroup(
        String problemID,
        String groupName);
  public void removeProblemFromGroup(
        String problemID,
        String groupName);
  public void addGroup(
        String groupName);
  public void addProblem(
        String problemID);
  public void addSubgroupRelationship(
        String groupName,
        String parentGroupName);
  public void addTriplet(
        String subject,
        String predicate,
        String object);
  public void deleteTriplet(
        String subject,
        String predicate,
        String object);
  public List queryTriplet(
        String subject,
        String predicate,
        String object);
}
```

Slika 7-12. Sučelje za potporu radu s ontologijama

Kako bi se upravljačima testova također olakšala uporaba semantičke mreže, StudTest sustav u okviru IReasoner sučelja definira StudTest-Zadaci ontologiju, opisanu u nastavku.

7.2.2 Ontologija StudTest-Zadaci

U ontologiji se definiraju koncepti/entiteti “type_group” i “type_problem”. Koncept “type_group” definira tip podatka koji predstavlja grupu koje sadrže zadatke i/ili druge grupe. Koncept “type_problem” definira tip podatka koji predstavlja zadatak.

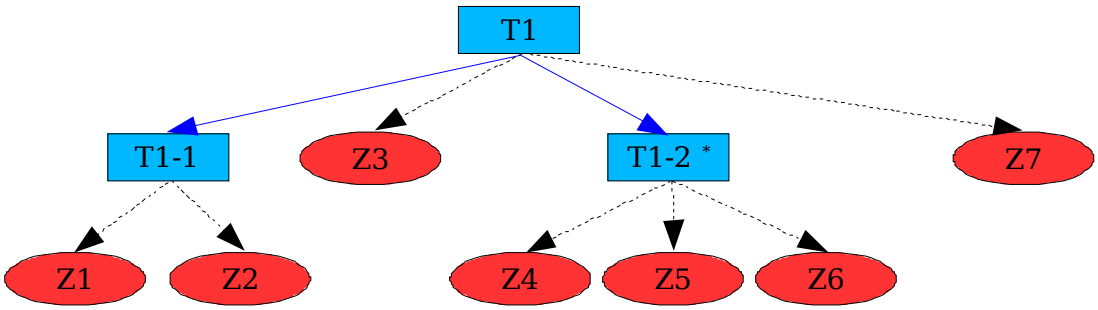
Definira se relacija “isOfType” čiji su subjekt i objekt neki entiteti. Izraz “X isOfType Y” definira X kao entitet koji je po tipu Y. Trenutno, u ovom kontekstu ontologija definira dvije moguće vrijednosti za Y: type_group i type_problem.

Definira se relacija “isSubgroupOf” čiji su subjekt i objekt entiteti tipa type_group. Izraz “X isSubgroupOf Y” definira da je X podgrupa od Y.

Definira se relacija “hasFlag” čiji su subjekt i predikat neki entiteti. Relacija “X hasFlag Y” definira da entitet X ima zastavicu Y. Ontologija za vrijednost Y trenutno definira samo mutuallyExclusiveMembers (što u kontekstu grupe znači da se iz grupe smije odabrati samo jedan zadatak).

Definira se relacija “isMemberOf” čiji je subjekt entitet tipa type_problem, a objekt entitet tipa type_group.

Ova jednostavna ontologija omogućava razmještanje zadataka u grupe i podgrupe, od kojih neke grupe mogu biti isključive. Slika 7-13 prikazuje primjer jedne takve organizacije, gdje je razmješteno 7 zadataka.



Slika 7-13. Primjer hijerarhijskog razmještanja zadataka

U primjeru je definirana grupa T1, koja sadrži podgrupu T1-1, zadatak Z3, isključivu podgrupu T1-2 te zadatak Z7. Podgrupa T1-1 sadrži zadatke Z1 i Z2, a podgrupa T1-2 sadrži zadatke Z4, Z5 i Z6. Slika 7-14 prikazuje ovaj primjer opisan ontologijom StudTest-Zadaci.

7.2.3 Uporaba ontologije StudTest-Zadaci

Prilikom stvaranja novih zadataka u StudTest sustavu, autor zadataka ima mogućnost zadatke opisati ontologijom StudTest-Zadaci. Prethodno opisani upravljač provjere znanja SimpleTestController već koristi ovakav opis za odabir zadataka koji mu stoje na raspolaganju. Slika 7-15 prikazuje isječak koda koji to ilustrira.

U kodu se jasno mogu raspoznati sljedeći koraci:

- stvaranje prazne liste zadataka

```
<”T1”,”isOfType”,”type_group”>
<”T1-1”,”isOfType”,”type_group”>
<”T1-1”,”isSubgroupOf”,”T1”>
<”T1-2”,”isOfType”,”type_group”>
<”T1-2”,”hasFlag”,”mutuallyExclusiveMembers”>
<”T1-2”,”isSubgroupOf”,”T1”>
<”Z1”,”isOfType”,”type_problem”>
<”Z1”,”isMemberOf”,”T1-1”>
<”Z2”,”isOfType”,”type_problem”>
<”Z2”,”isMemberOf”,”T1-1”>
<”Z3”,”isOfType”,”type_problem”>
<”Z3”,”isMemberOf”,”T1”>
<”Z4”,”isOfType”,”type_problem”>
<”Z4”,”isMemberOf”,”T1-2”>
<”Z5”,”isOfType”,”type_problem”>
<”Z5”,”isMemberOf”,”T1-2”>
<”Z6”,”isOfType”,”type_problem”>
<”Z6”,”isMemberOf”,”T1-2”>
<”Z7”,”isOfType”,”type_problem”>
<”Z7”,”isMemberOf”,”T1”>
```

Slika 7-14. Primjer opisa hijerarhije zadataka uporabom ontologije StudTest-Zadaci

- dohvat svih zadataka koji pripadaju zadanoj grupi, pri čemu se uključuju i zadaci podgrupa, jer se zahtjeva tranzitivno slijeđenje veza
- stvaranje generatora slučajnih brojeva
- petlju koja se izvodi tako dugo dok se ne odabere potreban broj zadataka, i koja u svakom prolazu
    - iz popisa raspoloživih zadataka odabere jedan zadatak, i doda ga u listu odabranih zadataka
    - ukloni taj zadatak iz liste odabranih zadataka
    - uporabom objekta `reasoner` dohvati sve zadatke koji su isključivi s odabranim zadatkom
    - iz popisa raspoloživih zadataka ukloni sve isključive zadatke

Prikazani kod samo je ilustracija stvarne implementacije, jer se konkretne implementacije mora nositi i s problemima nestašice zadataka (dojaviti pogrešku, ili nešto slično).

U ovom primjeru ontologija i uneseni opisi zadataka zapravo se uopće ne koriste na “inteligentan” način, već kao običan filtar, koji iz kolekcije svih zadataka koji se u određenom trenutku nalaze u poslužitelju filtrira one koji pripadaju određenoj grupi, te prilikom odabira samih zadataka pazi na zastavicu isključivosti.

Sustav StudTest predstavlja izvrsnu osnovu i za ugradnju prave umjetne inteligencije (odnosno odgovarajućih tehnika). Međutim, to će doći do izražaja tek razradom prilagodljivog vođenja ispitivanja.

```
List zadaci = new ArrayList();
List allProblems =
     reasoner.listProblemsInGroup(group,true);
Random rand = new Random();
int i = 0;
do {
  int total = allProblems.size();
  if(total<1) break;
  int odabrano = rand.nextInt(total);
  String maskedProblemID =
     (String)allProblems.get(odabrano);
  allProblems.remove(odabrano);
  String[] parts = maskedProblemID.split(":");
  String problemID = parts[1];
  zadaci.add(problemID);
  i++;
  List excl =
     reasoner.listProblemsExclusiveWith(maskedProblemID);
  if(excl==null || excl.size()==0) continue;
  allProblems.removeAll(excl);
} while(i < brojZadatakaZaStvoriti);
```

Slika 7-15. Primjer uporabe ontologije za odabir zadataka

8 Primjeri razvijenih zadataka

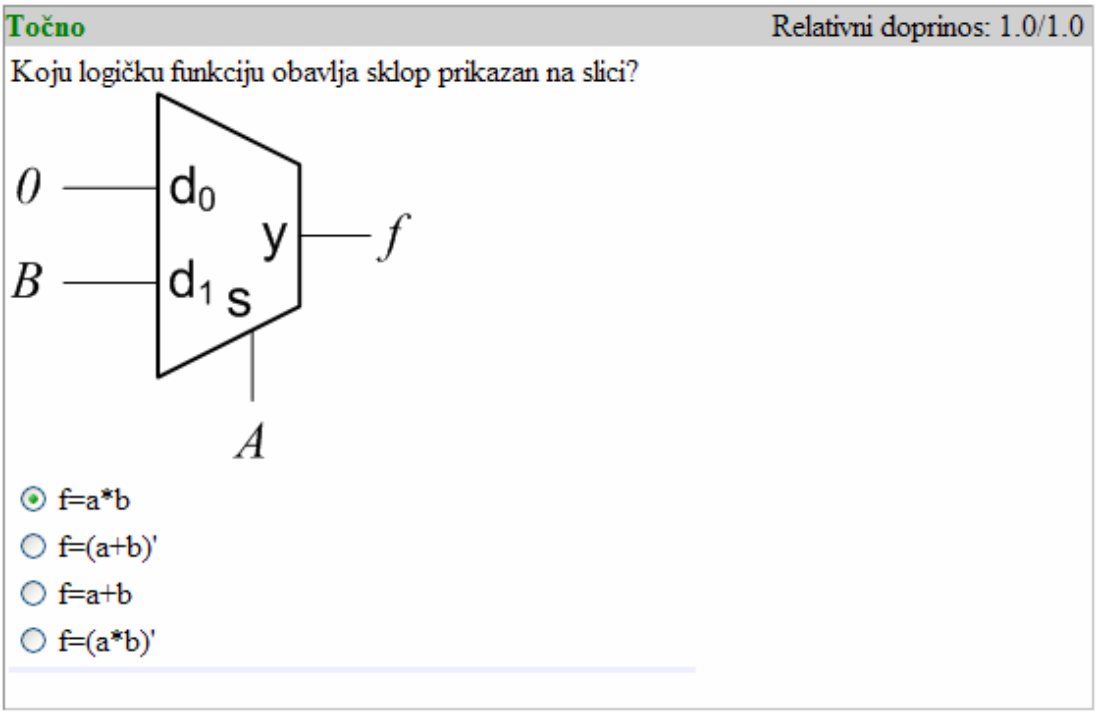
Kao prikaz mogućnosti sustava temeljenih na modelu StudTest, u nastavku će biti prikazano nekoliko stvarnih primjera zadataka, korištenih na kolegijima Digitalna elektronika i Digitalna logika.

8.1 Statičke ABC-pitalice

Statičke ABC pitalice su zadaci kod kojih se pitanje i popis odgovora zadaje unaprijed, te nema nikakvih elemenata dinamičnosti.

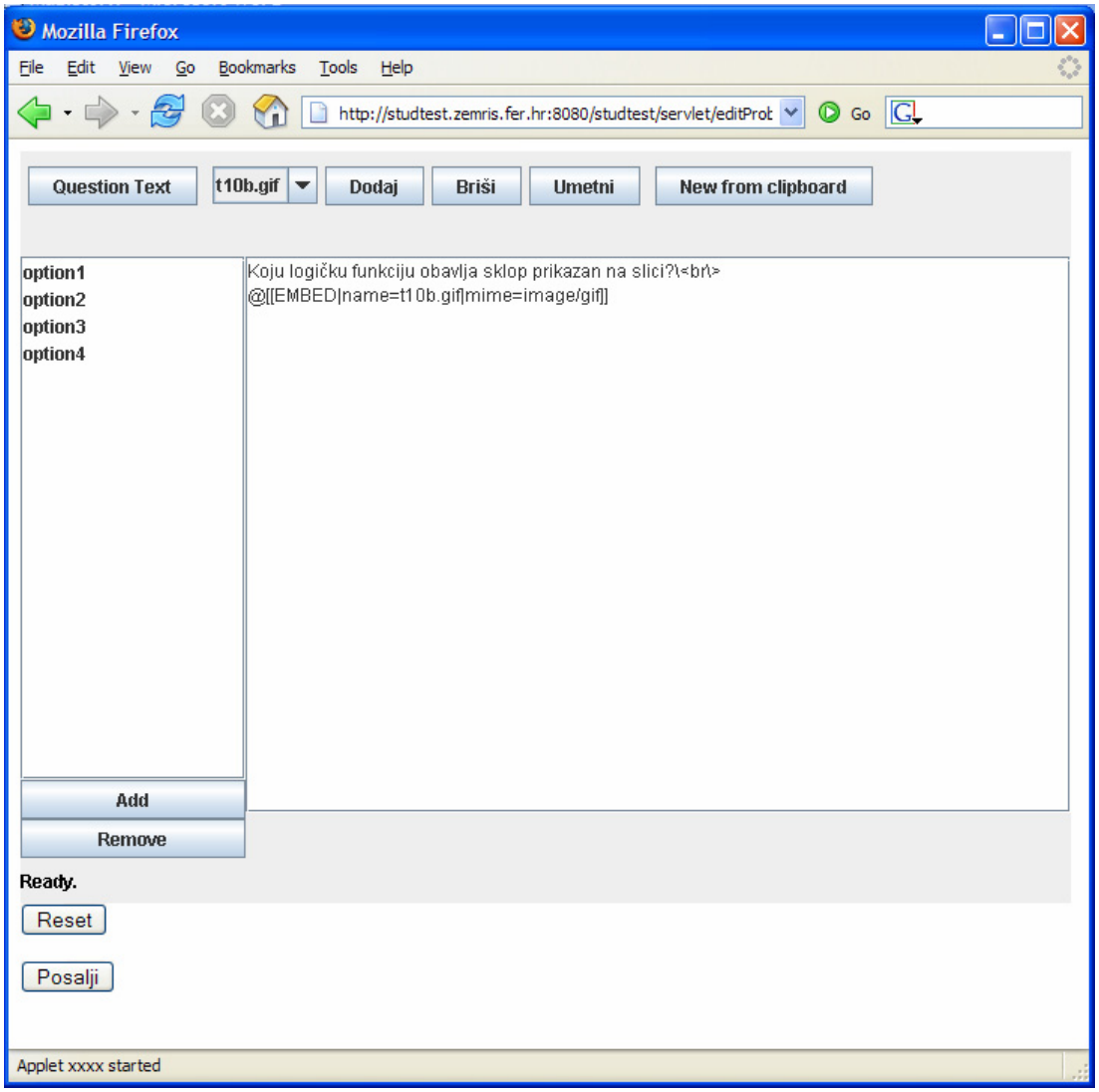
8.1.1 Statička pitanja s jednim točnim odgovorom

Primjer jednog ovakvog pitanja (u obliku koji vidi ispitanik) prikazan je sljedećom slikom. Slika sklopa, pitanje i odgovori unaprijed su pripremljeni.



Slika 8-1. Primjer zadatka

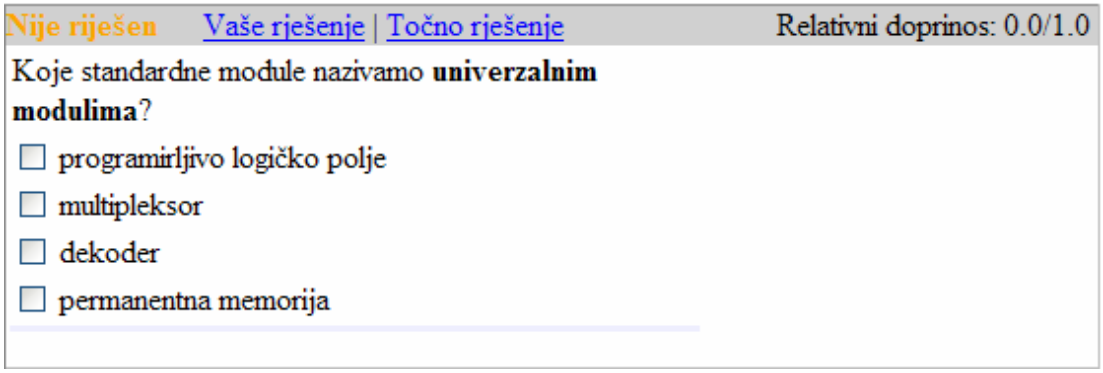
Uređivač ovakve vrste zadataka izveden je u okviru samo jednog generatora zadataka, i prikazan je na sljedećoj slici.



Slika 8-2. Primjer zadatka

8.1.2 Statička pitanja s više točnih odgovora

U ovom primjeru vidi se promjena načina na koji student unosi odgovor, gdje kontrola checkbox jasno upućuje na mogućnost više točnih opcija.



Slika 8-3. Primjer zadatka

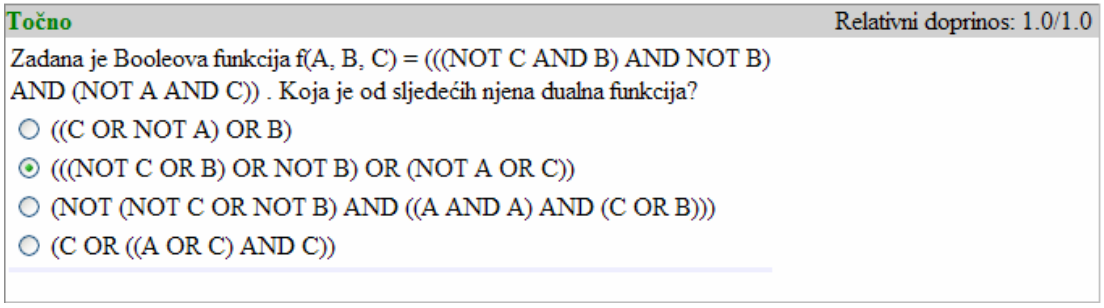
Uređivač ovakve vrste zadataka gotovo je identičan uređivaču prethodne vrste, pa se zbog uštede u prostoru neće ponavljati.

8.2 Dinamičke ABC-pitalice

Dinamičke ABC pitalice su zadaci kod kojih se tekst pitanja i popis ponuđenih odgovora generiraju dinamički, prilikom stvaranja samog pitanja.

8.2.1 Dualna funkcija

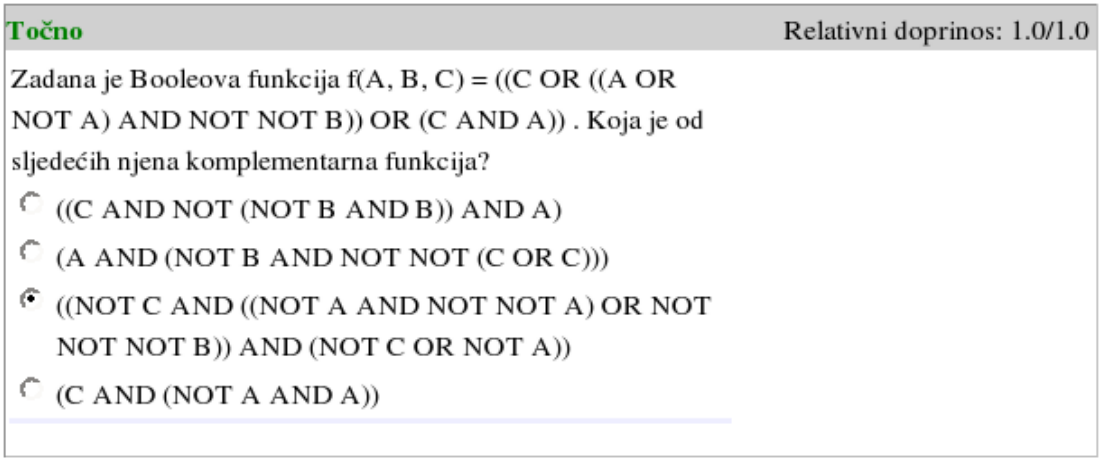
U ovom zadatku dinamički se generira zadana funkcija, kao i skup ponuđenih funkcija u odgovorima. Na ovaj način svaki student može dobiti potpuno različito pitanje, koje nije moguće prepisati.



Slika 8-4. Primjer zadatka

8.2.2 Komplementarna funkcija

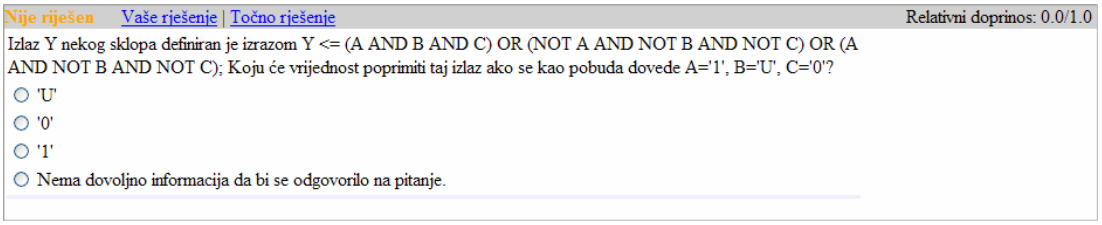
I u ovom zadatku dinamički se za svakog studenta generira funkcija *f*, i četiri potencijalna rješenja od kojih je jedno točno.



Slika 8-5. Primjer zadatka

8.2.3 Rezultat izračuna izraza jezika VHDL

U ovom zadatku dinamički se za svakog studenta stvara jedan izraz u jeziku VHDL, te student mora odgovoriti kakav će biti rezultat simulacije tog izraza uz određenu pobudu.



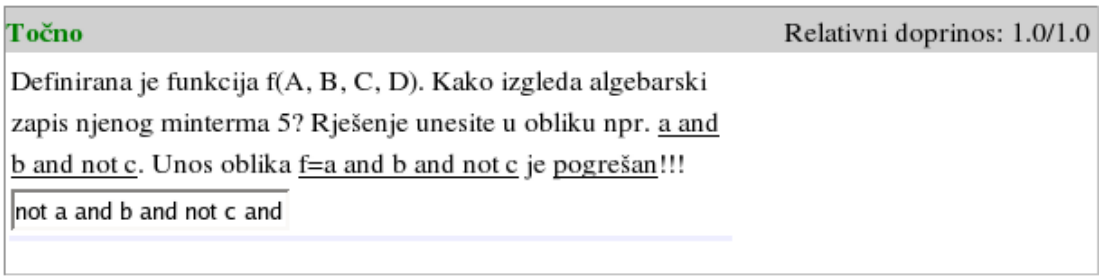
Slika 8-6. Primjer zadatka

8.3 Unos teksta

"Unos teksta" su zadaci na koje korisnik odgovara upisivanjem niza znakova.

8.3.1 Algebarski zapis minterma/maksterma

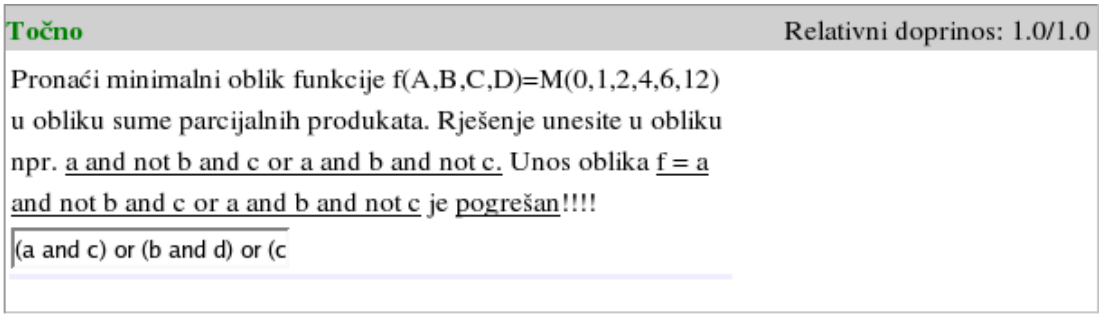
U ovom primjeru student treba unijeti algebarski zapis minterma funkcije od 4 varijable (broj varijabli i indeks minterma, odnosno želi li se minterma ili maksterm odabire se dinamički za svakog studenta). Vrednovatelj je u stanju obraditi uneseni izraz i provjeriti njegovu točnost neovisno o dozvoljenim transformacijama (npr. komutativnost).



Slika 8-7. Primjer zadatka

8.3.2 Minimizacija funkcije

U ovom primjeru za svakog se studenta dinamički stvori zadana funkcija, te student mora upisati jedan od minimalnih zapisa te funkcije, što vrednovatelj potom može provjeriti, opet neovisno o dozvoljenim transformacijama nad izrazom (poput komutativnosti).

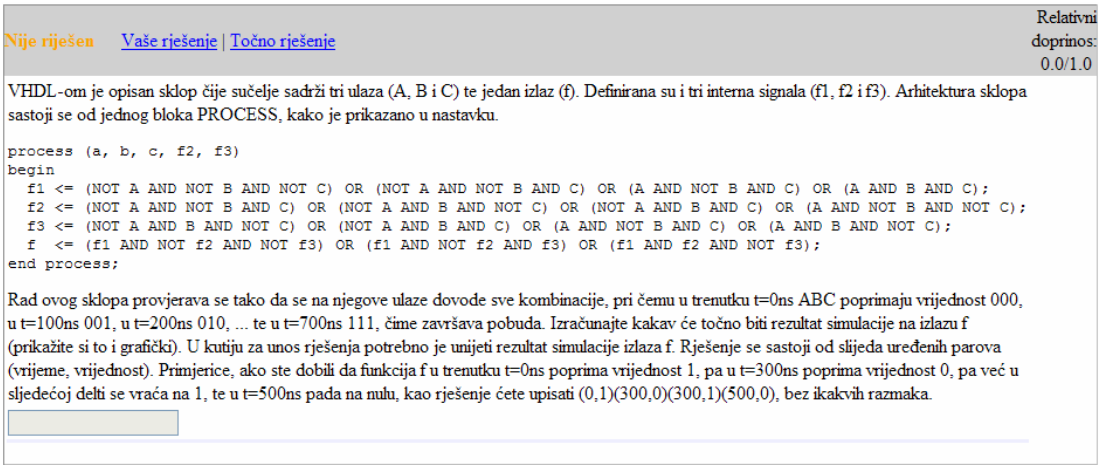


Slika 8-8. Primjer zadatka

8.3.3 Rezultat simulacije modela VHDL

Ovaj primjer predstavlja jedan vrlo kompleksan zadatak, jer se za svakog studenta dinamički generira opis jednog modela opisanog u jeziku VHDL, te student mora

odgovoriti kakav će biti rezultat simulacije tog sklopa. Primjer detaljno ispituje razumije li student načela rada simulatora modela napisanih u jeziku VHDL. Vrednovatelj zadatka je u stanju sam obaviti simulaciju rada sklopa, te provjeriti studentovo rješenje.


Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

VHDL-om je opisan sklop čije sučelje sadrži tri ulaza (A, B i C) te jedan izlaz (f). Definirana su i tri interna signala (f1, f2 i f3). Arhitektura sklopa sastoji se od jednog bloka PROCESS, kako je prikazano u nastavku.

```
process (a, b, c, f2, f3)
begin
  f1 <= (NOT A AND NOT B AND NOT C) OR (NOT A AND NOT B AND C) OR (A AND NOT B AND C) OR (A AND B AND C);
  f2 <= (NOT A AND NOT B AND C) OR (NOT A AND B AND NOT C) OR (NOT A AND B AND C) OR (A AND NOT B AND NOT C);
  f3 <= (NOT A AND B AND NOT C) OR (NOT A AND B AND C) OR (A AND NOT B AND C) OR (A AND B AND NOT C);
  f  <= (f1 AND NOT f2 AND NOT f3) OR (f1 AND NOT f2 AND f3) OR (f1 AND f2 AND NOT f3);
end process;
```

Rad ovog sklopa provjerava se tako da se na njegove ulaze dovode sve kombinacije, pri čemu u trenutku t=0ns ABC poprimaju vrijednost 000, u t=100ns 001, u t=200ns 010, ... te u t=700ns 111, čime završava pobuda. Izračunajte kakav će točno biti rezultat simulacije na izlazu f (prikažite si to i grafički). U kutiju za unos rješenja potrebno je unijeti rezultat simulacije izlaza f. Rješenje se sastoji od slijeda uređenih parova (vrijeme, vrijednost). Primjerice, ako ste dobili da funkcija f u trenutku t=0ns poprima vrijednost 1, pa u t=300ns poprima vrijednost 0, pa već u sljedećoj delti se vraća na 1, te u t=500ns pada na nulu, kao rješenje ćete upisati (0,1)(300,0)(300,1)(500,0), bez ikakvih razmaka.

Slika 8-9. Primjer zadatka

8.4 Unos liste

"Unos liste" su zadaci kod kojih korisnik u sklopu odgovaranja upisuje više tekstualnih odgovora.

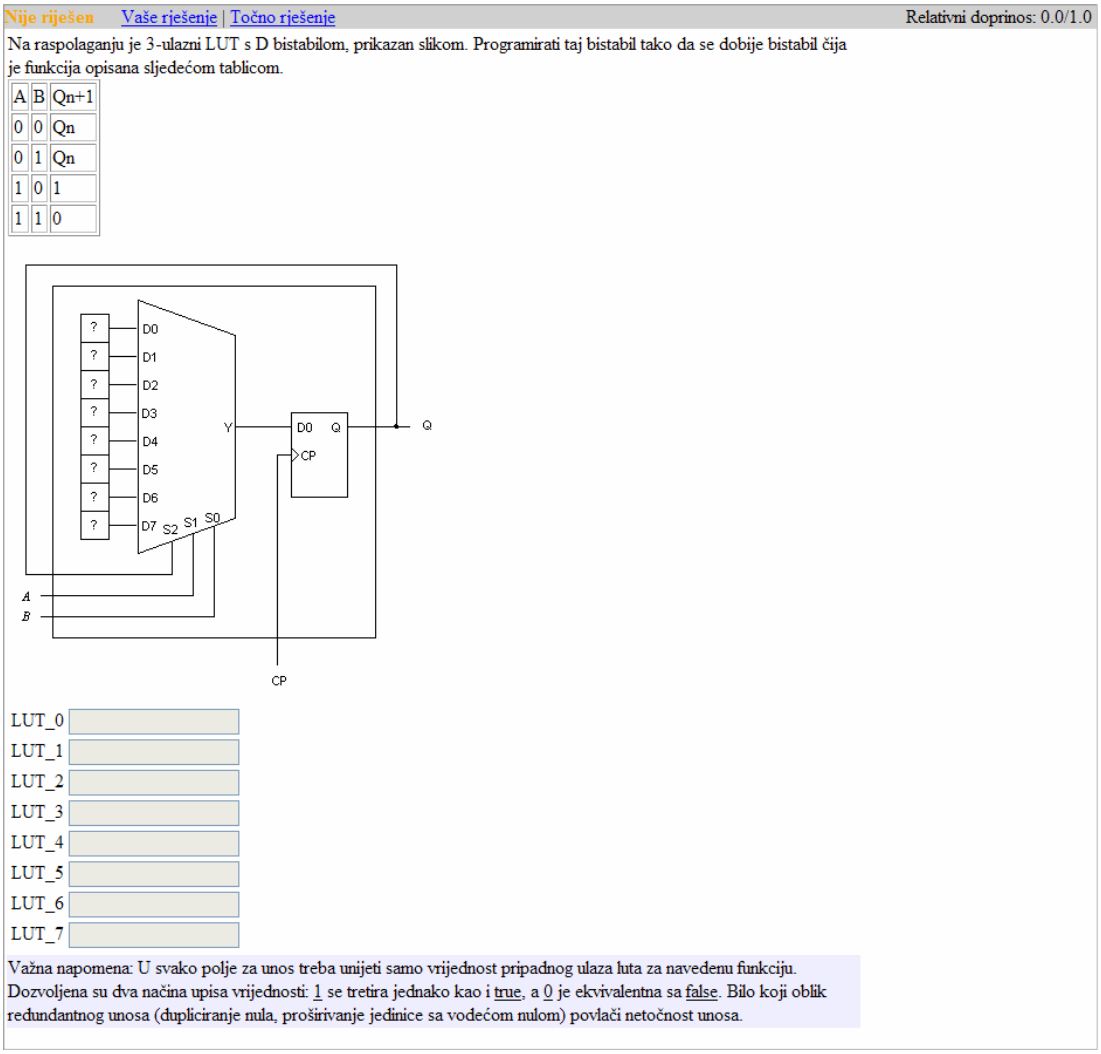
8.4.1 Minimizacija Quine-McClusky

U ovom primjeru studentu se dinamički generira Booleova funkcija, koju student potom mora minimizirati uporabom postupka Quine-McCluskey. Kao rješenje, student upisuje konačni odgovor (sve minimalne oblike) i međurezultate (sve primarne implikante, te sve bitne primarne implikante), što vrednovatelj može provjeriti.

Slika 8-10. Primjer zadatka

8.4.2 Programiranje sekvencijskog sklopa preglednim tablicama

U ovom primjeru student ima zadatak uporabom jednog logičkog bloka FPGA sklopa realizirati bistabil, čija je tablica promjene stanja slučajno generirana. Kako se i veličina LUT-a generira dinamički (i u skladu s odabranim brojem ulaza bistabila), ovaj zadatak također nudi vrlo veliki broj kombinacija različitih pitanja, čime pojavu prepisivanja efikasno eliminira.



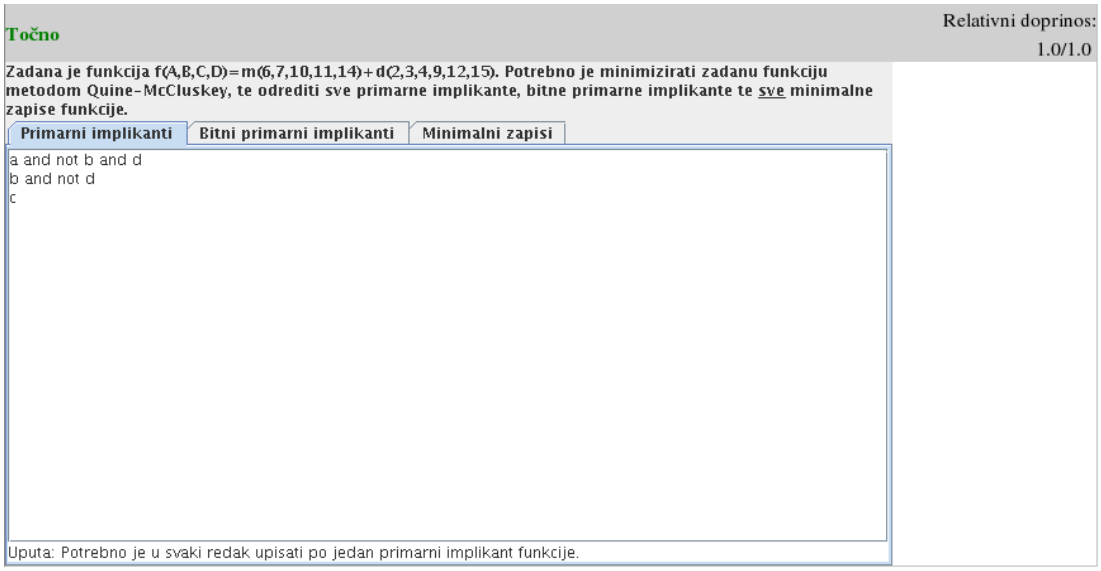
Slika 8-11. Primjer zadatka

8.5 Pitanja složenog grafičkog korisničkog sučelja

Ova vrsta predstavlja pitanja gdje korisnici za potrebe odgovaranja koriste tehnologiju Java Appleta.

8.5.1 Minimizacija Quine-McClusky

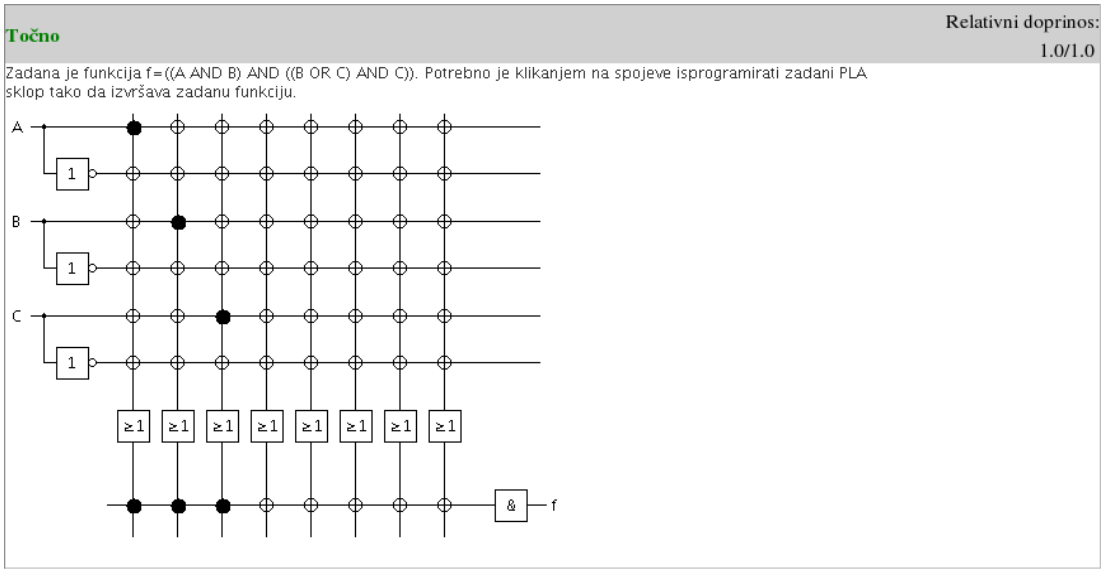
U ovom primjeru za studenta se dinamički stvara Booleova funkcija koju je potrebno minimizirati uporabom postupka Quine-McCluskey. No za razliku od već navedenog sličnog primjera (poglavlje 8.4.1), ovdje se unos rezultata obavlja kroz grafičko korisničko sučelje uporabom tehnologije Java Applet, odnosno kontrole JTabbedPane.



Slika 8-12. Primjer zadatka

8.5.2 Programiranje sklopa PLA

U ovom primjeru za svakog se studenta generira nova funkcija koju je potrebno realizirati uporabom PLA sklopa. Student odgovara na zadatak tako što mišem označava spojeve koje treba ostvariti kako bi se na izlazu dobila željena funkcija. Interakcija s korisnikom također se odvija kroz tehnologiju Java Applet.

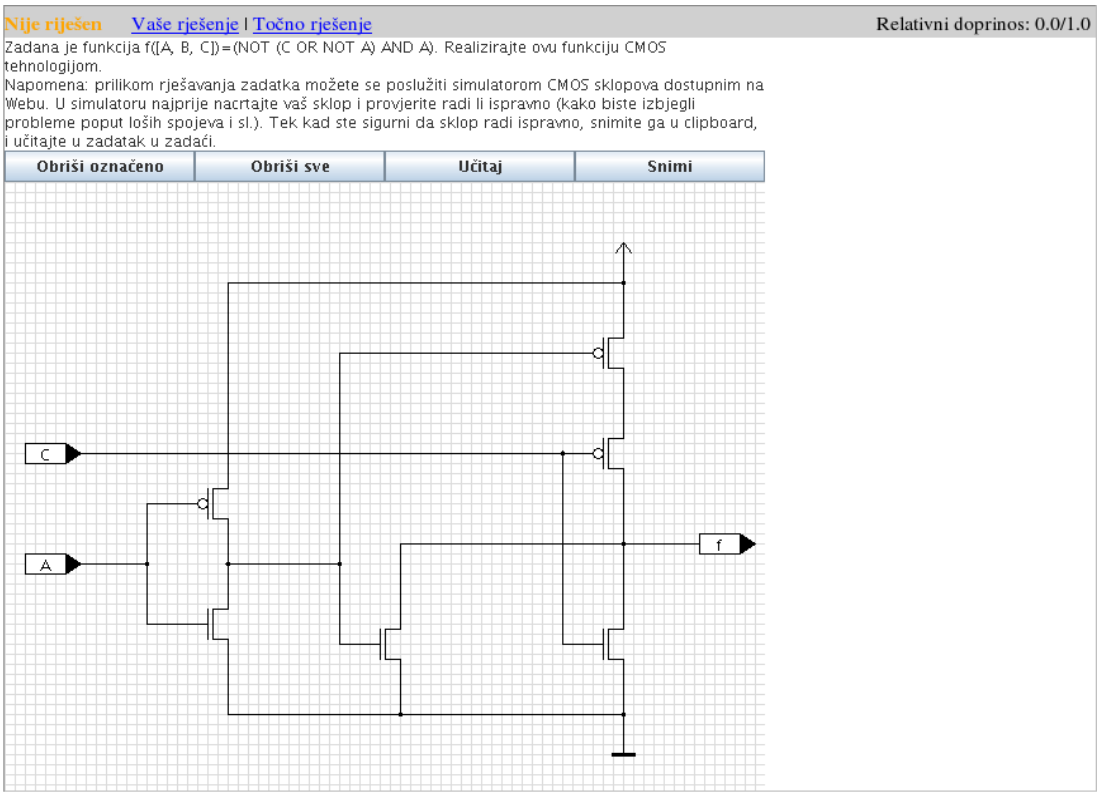


Slika 8-13. Primjer zadatka

8.5.3 Implementacija funkcije tehnologijom CMOS

Ovo je jedan od najsloženijih implementiranih zadataka. U ovom zadatku, studentu se slučajno generira neka Booleova funkcija, koju je potom potrebno realizirati uporabom tehnologije CMOS. Kako bi riješio zadatak, student kroz sučelje koje nudi tehnologija Java Applet (i u njoj implementirano okruženje za crtanje sklopova) mora nacrtati shemu sklopa uporabom P-kanalnih i N-kanalnih tranzistora. Za potrebe vrednovatelja ovog zadatka napisan je simulator koji je u mogućnosti

odsimulirati rad nacrtanog sklopa, i provjeriti je li on u skladu sa zadanom funkcijom.



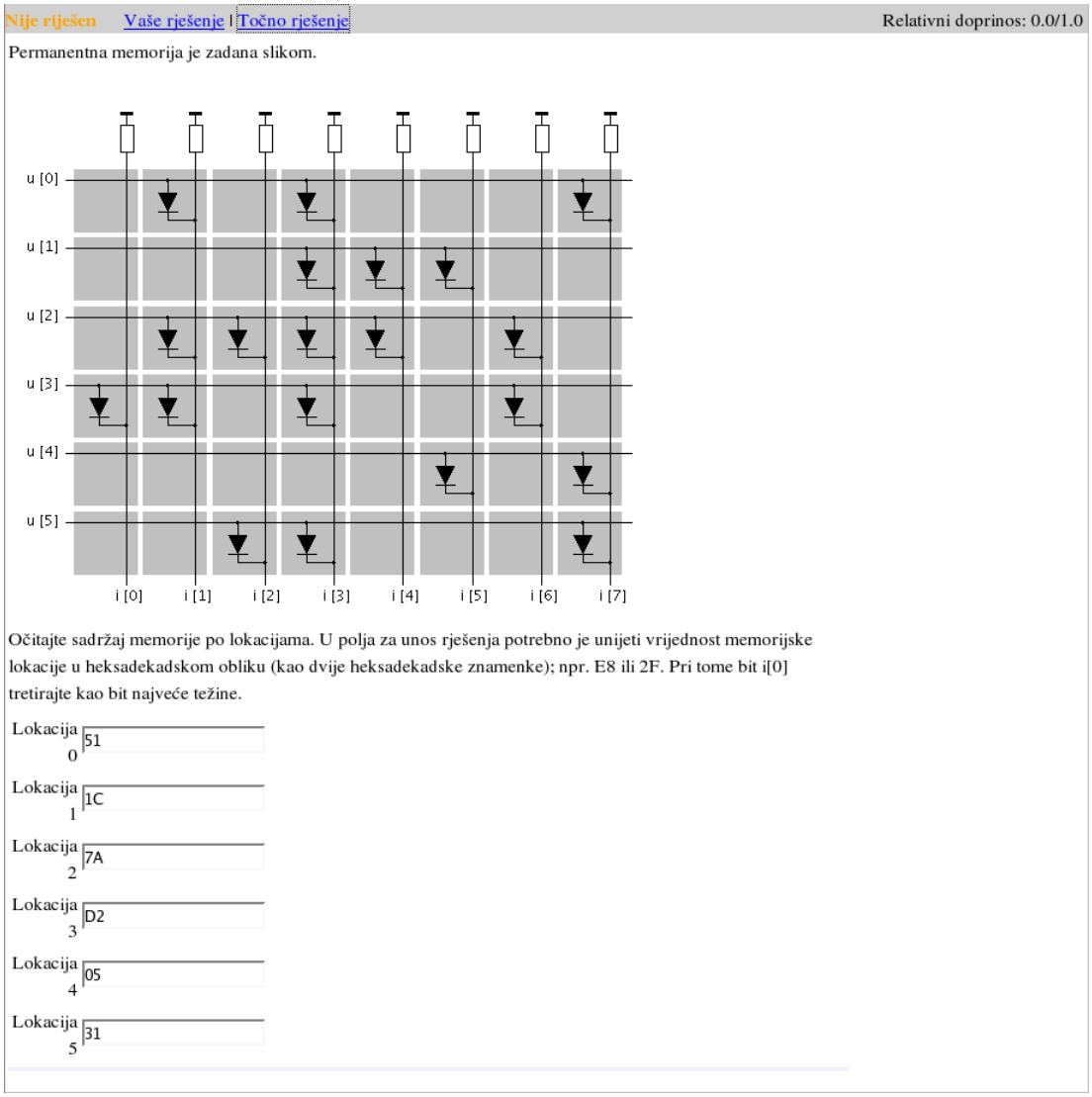
Slika 8-14. Primjer zadatka

8.6 Uporaba pomoćnika

Svi dinamički generirani zadaci, neovisno o tipu, za potrebe postavljanja pitanja ili nuđenja odgovora mogu u zadatak ubacivati elemente višemedijskih sadržaja koji će se generirati u letu, ovisno o parametrima primjerka zadatka konkretnog korisnika. U nastavku su prikazana dva implementirana zadatka.

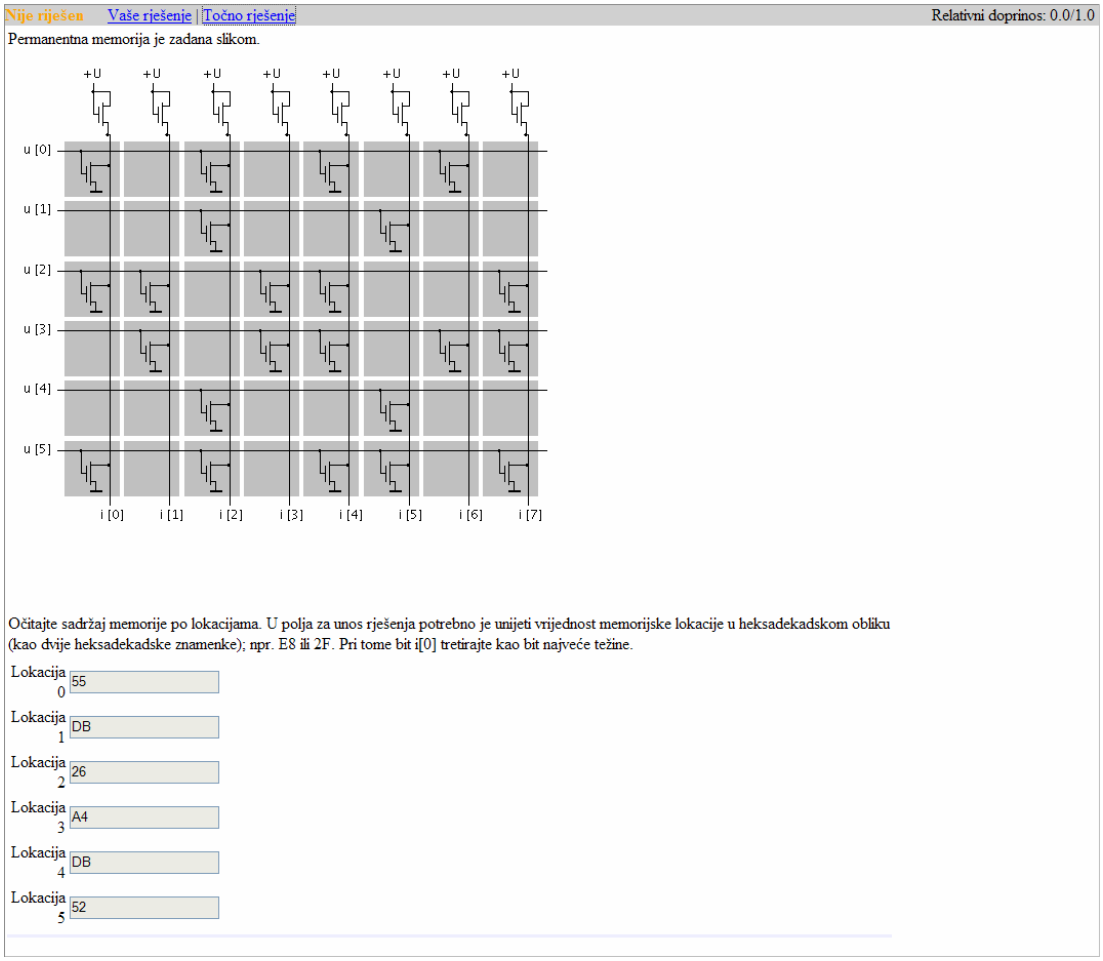
8.6.1 Očitavanje sadržaja memorije izvedene diodama i tehnologijom MOSFET

Ovo je primjer zadatka koji koristi usluge pomoćnika za dinamičko stvaranje slike koja prikazuje način izvedbe memorije (gdje su podržane dvije izvedbe: izvedba diodnim poljem, te izvedba MOSFET-ima). Prilikom stvaranja zadatka, za svakog se studenta dinamički generira sadržaj memorije koji se posprema u repozitorij primjerka zadatka. U trenutku kada student zatraži svoj zadatak (odnosno kada se zadatak mora prikazati, poziva se pomoćnik koji u letu na temelju sadržaja memorije zapisanog u repozitoriju primjerka zadatka nacrta sliku memorije, i dostavi je klijentu).



Slika 8-15. Primjer zadatka

U prethodnom primjeru prikazan je zadatak u kojem je memorija ostvarena diodnim poljem. U sljedećem primjeru, memorija je realizirana tehnologijom MOSFET.



Slika 8-16. Primjer zadatka

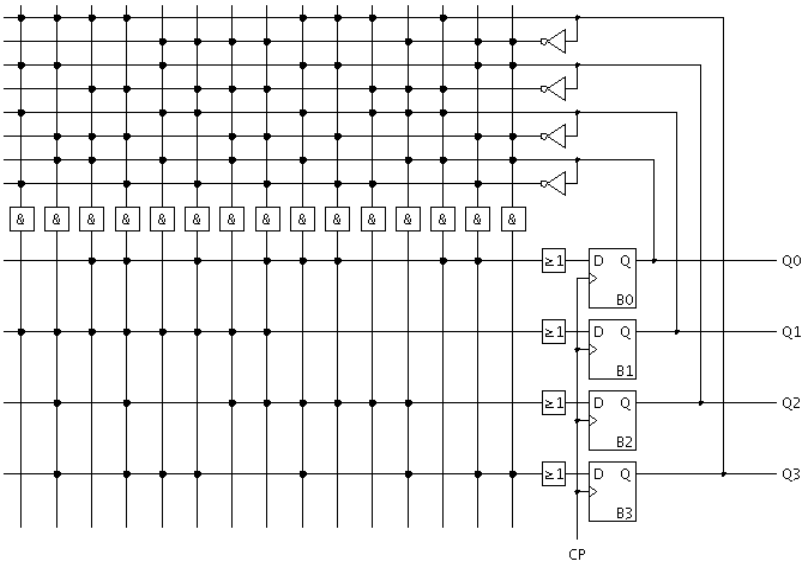
8.6.2 Očitavanje ciklusa sekvencijskog sklopa

Ovo je također primjer uporabe pomoćnika koji se koristi za crtanje sekvencijskog sklopa. Prilikom stvaranja zadatka, najprije se odabire željeni broj bistabila i vrsta bistabila. Potom se na temelju broja bistabila dinamički odabire proizvoljni ciklus, te se obavlja postupak projektiranja takvog sklopa uporabom odabranih bistabila. Rezultati projektiranja sklopa pohranjuju se u repozitorij primjerka zadatka. U trenutku prikaza zadatka studentu, poziva se pomoćnik koji dohvaća te rezultate, i na temelju njih u letu stvara sliku odgovarajućeg sklopa, koja se potom prikazuje studentu.

Nije riješen Vaše rješenje | Točno rješenje

Relativni doprinos: 0.0/1.0

Neko sinkrono brojilo prikazano je sljedećom slikom.



U kojem ciklusu broji to brojilo? Prilikom očitavanja stanja izlaz Q0 tretirati kao bit najmanje težine.

- (14, 8, 7, 15, 11, 3, 5, 13, 2, 0, 12, 10, 6, 4, 1, 9, 14)
- (4, 7, 5, 14, 3, 12, 0, 9, 2, 13, 10, 6, 8, 15, 11, 1, 4)
- (7, 12, 3, 0, 4, 10, 13, 8, 2, 5, 14, 1, 9, 6, 11, 15, 7)
- (9, 3, 12, 5, 8, 15, 13, 14, 2, 11, 1, 6, 0, 7, 10, 4, 9)

Slika 8-17. Primjer zadatka

9 Zaključak

Ovaj rad opisuje model sustava e-ispitivanja StudTest. U radu je prikazan model sustava e-ispitivanja koji je nastao na temelju analize zahtjeva niza kolegija u pogledu ispitivanja studenata elektroničkim putem, a koje je naročito interesantno na masovnim kolegijima koje pohađaju stotine studenata, odnosno čak i više od 1000 studenata. Pri tome, sustav podržava vođenje kako ispita u strogom smislu, tako i domaćih zadaća, koje se mogu shvatiti kao ispiti koji se ne pišu pod strogim nadzorom asistenata, i kojima se može pristupati višestruko, prije no što se sama zadaća vrednuje i ocijeni.

Također, prikazana je prototipna implementacija sustava, koja se u nastavnom procesu uspješno koristi već dvije godine, u okviru kolegija na Fakultetu elektrotehnike i računarstva, i to na kolegiju Digitalna elektronika (po programu FER 1), odnosno jednu godinu na kolegiju Digitalna logika (po programu FER 2 u skladu s Bolonjskim procesom, u zimskom semestru, i na ponovljenom kolegiju u ljetnom semestru akademske godine 2005/2006). Tako je, primjerice, na kolegiju Digitalna logika u zimskom semestru na sustavu radilo preko 1100 studenata. Sustav se je koristio za pisanje izlaznih testova na laboratorijskim vježbama, te za izradu domaćih zadaća. Pri tome je tijekom cijelog semestra sustav studentima ukupno postavio više od 115 000 pitanja. Gledano po zadacima, u sustav je uneseno više od 60 statičkih pitanja, te više od 60 dinamičkih pitanja korištenih u domaćim zadaćama, koja su rezultirala personaliziranim domaćim zadaćama koje studenti nisu mogli prepisivati.

Usporedbom sa sustavima poput WebCT i MOODLE može se zaključiti kako je razvijeni sustav bolje rješenje za e-ispitivanje, jer navedeni sustavi gotovo da i nemaju potporu za izradu dinamičkih zadataka, što je glavna prednost koju nudi ispitivanje putem računala.

Također, u radu je prikazan način na koji se u opisanom sustavu može razraditi inteligentno vođenje ispitnog procesa, što je svakako jedan od sljedećih koraka u razvoju sustava. Naime, jednom kada se kvalitetno riješi inteligentno vođenje ispitnog procesa, ovakav sustav može poslužiti za izradu vrlo kvalitetnog alata za samostalno učenje (ili učenje na daljinu), gdje bi student učio određeno gradivo po dostupnim materijalima, te koristio sustav za provjeru stečenog znanja. Pri tome se može otići i korak dalje, kao što je prikazano u radu, i definirati strukturu gradiva koje je potrebno naučiti, dostupne materijale vezane uz određene granule znanja (što mogu biti i objekti SCO), te popis pitanja vezanih uz te granule znanja. Po obavljenoj provjeri znanja, uz takav se opis student može točno uputiti da prouči one materijale koji pojašnjavaju i razrađuju nesavladano gradivo.

Tijekom rada s prototipnom implementacijom javilo se je i nekoliko praktičnih problema, koji ukazuju na pravce daljnjeg razvoja. Tako sustav ne podržava definiranje parametara provjere znanja za svakog studenta posebno (primjerice, molba stranog studenta koji slabije razumije hrvatski jezik za dužim vremenom pisanja provjera nije mogla biti ispunjena). Uočena je i potreba za potporom provjere sintakse unesenog rješenja uživo (prije bodovanja), kako bi se ispitanik upozorio na eventualne tipografske pogreške, te dinamičko generiranje pomoći, što je korisno u situacijama gdje student ne zna kako nastaviti s rješavanjem nekog zadatka. Tada mu

generator zadatka može dinamički stvoriti uputu što je potrebno napraviti (ali zbog toga će i maksimalni broj bodova biti smanjen).

Zanimljivo je također istaknuti pojavu potrebe za novim tipom zadataka – zadataka sa stanjem, koji se rješavaju u više koraka. Ovu vrstu zadataka možemo shvatiti kao bilježnicu: na svakom listu piše jedan dio zadatka (podzadatak). Točnim odgovorom prelazi se dalje ("na sljedeći list").

Iz svega navedenoga može se zaključiti da je ovakav pristup sustavu e-ispitivanja ispravan, o čemu svjedoči i njegova višegodišnja uspješna primjena.

Bibliografija

1 Williams, M., *E-Learning: History and prospects, Education without borders 2005*, Newsletter http://e-ducation2005.com/ewb_newsletter_04_10.pdf

2 Bouras, C., Hornig, G., Triantafillou, V., Tsiatsos, T. (2001) "Architectures Supporting e-Learning Through Collaborative Virtual Environments: The Case of INVITE", *IEEE International Conference on Advanced Learning Technologies-ICALT 2001*, Madison, Wisconsin, USA, 6 - 8 August 2001, pp. 13 – 16

3 Georgieva, E., Smrikarov, A., Georgiev, T., "A General Classification of Mobile Learning Systems", Proc. Intl Conference on Computer Systems and Technologies - CompSysTech' 2005, pp. IV.14-1 – IV.14-6

4 Eduworks Corporation, *The Global Framework for E-learning* http://www.eduworks.com/Documents/Presentations/AICTEC_2003_GlobalFramework.ppt

5 Gettinger, M. (1984) Individual differences in time needed for learning: A review of the literature. *Educational Psychologist*, 19,15-29.

6 Graesser, A. C., Person, N. K. (1994). Question asking during tutoring. *American Educational Research Journal*, 31, 104-137.

7 Bloom, B.S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13, 4-16.

8 Stankov, S. (1998) Intelligent Tutoring Knowledge Base Authoring Shell: Tutor-Expert System (TEx-Sys), *Journal of electrical engineering* : Czech-Slovak section of IEEE. 49 (1998) , 5-6; 149-155

9 Liu, X., El Saddik A., Georganas, N. D. (2003) An Implementable Architecture of an E-Learning System. Electrical and Computer Engineering, *IEEE CCECE 2003. Canadian Conference*, pp. 717 - 720 vol.2

10 Mavromoustakos, S. M., Papanikolaou, K., Andreou, A. S., Papadopoulos, G. A. (2005) "Human and Technological Issues in the E-Learning Engineering Process", *International Conference on E-Business and E-Learning – EBEL05*, May 23-24, Amman, Jordan

11 *SUN: E-learning framework* http://www.sun.com/products-n-solutions/edu/whitepapers/pdf/framework.pdf

12 *IBM: Lotus LearningSpace* http://www-306.ibm.com/software/lotus/support/learningspace/support.html

13 Learning Technologies Standardization Committee (LTSC). http://ltsc.ieee.org/

14 Dublin Core Metadata Initiative (DCMI). dublincore.org/

15 IMS Global Learning Consortium. http://www.imsproject.org/

16 US Department of Defense, Advanced Distributed Learning (ADL) Initiative.
http://www.adlnet.org/
17 Schools Interoperability Framework (SIF)
http://www.sifinfo.org/
18 Specifikacija *IMS Question and Test Interoperability*
http://www.imsglobal.org/question/qti_v2p0/imsqti_oviewv2p0.html
19 The ARIADNE Foundation
http://www.ariadne-eu.org/
20 *LOM Learning Object Metadata IEEE 1484.12 standard draft*
http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
21 Groš, S. (2004) *"Sustav obrazovanja na daljinu zasnovan na arhitekturi temeljenoj na uslugama"*, Magistarski rad, Zagreb, Fakultet elektrotehnike i računarstva, Voditelj: Vlado Glavinić
22 *IMS Question and Test Interoperability Overview*
http://www.imsglobal.org/question/qti_v2p0/imsqti_oviewv2p0.html
23 *IMS Question and Test Interoperability Implementation Guide*
Version 2.0 Final Specification
http://www.imsglobal.org/question/qti_v2p0/imsqti_implv2p0.html
24 Glavinić, V., Čupić, M., Groš, S. (2005) Nescume - A System for Managing Student Assignments, *Proceedings of the First International Conference on Internet Technologies and Applications (ITA 05)* / Grout, Vic ; Oram, Denise ; Picking, Rich (ur.)., Wrexham, North Wales, UK : Centre for Applied Internet Research (CAIR), University of Wales, NEWI, pp. 233-238
25 Glavinić, V., Čupić, M., Groš, S. (2004) WODLS - A Web Oriented Distance Learning System, *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference - MELECON 2004*, Volume II / Matijasevic, Maja ; Pejcinovic, Branimir ; Tomsic, Zeljko ; Butkovic, Zeljko (ur.)., Zagreb : The Institute of Electrical and Electronics Engineers, Inc., pp. 747-750
26 IBM Lotus Software
http://www.ibm.com/software/lotus
27 *Java 2 Platform EE v1.3: Interface Servlet*
http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/servlet/Servlet.html
28 *IETF RFC 2048:1996 Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures, 1996-11*,
dostupno na “http://www.ietf.org/rfc/rfc2048.txt”
29 Gruber, T. (1994) Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *IJHCS*, 43(5/6): 907-928.
30 Guarino, N. (1997) “Understanding, building and using ontologies”, *Int. Journal Human-Computer Studies*, v. 45, n. 2/3, (Feb/Mar)
31 Woods, W. A. (1975) What's in a Link: Foundation for Semantic Networks, Representation and Understanding Studies in Cognitive Science, D. Bobrow, A. Collins (eds.), pp. 35-82

32 Brewster, C., O'Hara, K., Fuller, S., Wilks, Y., Franconi, E., Musen, M.A., Ellman, J., Buckinham Shum, S. (2004) Knowledge Representation with Ontologies: The Present and Future. *IEEE Intelligent Systems* vol 19(1), pp. 72-81

33 Baader, F., Calvanese, F., McGuinness, D., Nardi, D., Patel-Schneider, P. editors (2002) *The Description Logic Handbook*. Cambridge University Press

34 Čupić, M. (2002) "*Inteligentno pretraživanje informacijskog prostora informacijske infrastrukture*", Diplomski rad, Zagreb, Fakultet elektrotehnike i računarstva, Voditelj: Vlado Glavinić

35 Resource Description Framework (RDF) / W3C Semantic Web Activity http://www.w3.org/RDF/

36 W3C Semantic Web http://www.w3.org/2001/sw/

37 Java Applets http://java.sun.com/applets/

38 Hibernate http://www.hibernate.org/

39 MySQL http://dev.mysql.com/

40 LabView – The Software That Powers Virtual Instrumentation http://www.ni.com/labview/

41 Dalbelo Bašić, B., Glavinić, V., Čupić, M. (2003) Simulation Supported Learning of Soft Computing Models, Human Computer Interaction: Theory and Practice (Part II), Volume 2 of *Proc. HCI International 2003* / Stephanidis, Constantine ; Jacko, Julie (ur.)., London : Lawrence Earlbaum Associates, pp. 1076-1080

42 Čupić, M., Šnajder, J., Dalbelo Bašić, B. (2003) Educational Interactive Software as a Support to the Teaching of Artifical Neural Network Methodology Applied to a Classification Problem, *Advances in technology-based education: Toward a knowledge-based society*, Badajoz, Spain, 2003. 1975-1979

43 FSCELi – FindS, CA, ID3 http://www.zemris.fer.hr/predmeti/su/java/FSCEli/FSCEli.html

44 WebCT http://www.webct.com/

45 MOODLE http://moodle.org/

46 Berners-Lee, T., Hendler, J. and Lassila, O. (2001) 'The Semantic Web, *Scientific American*', Vol. 5, pp.34–43.

47 Fensel, D., Wahlster, W., Lieberman, H. (2002) *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, MIT Press Cambridge, MA, USA

48 Antoniou, G., Van Harmelen F. (2004) *A Semantic Web Primer (Cooperative Information Systems)*, MIT Press, Cambridge, MA, USA

10 Prilozi

10.1 Konstante i pomoćni objekti modela StudTest

U ovom odjeljku slijedi definicija i opis konstanti važnih za osnovne koncepte modela, kao i definicije pomoćnih objekata.

Tablica 10-1. Vrijednosti elementa TestInstance.status

Naziv konstante	Opis konstante
TST_UNSOLVED	Početno stanje.
TST_PREPARED	Provjera je pripremljena.
TST_PROHIBITED	Provjera je zabranjena.
TST_SOLVING_IN_PROGRESS	Rješavanje provjere je u tijeku.
TST_SOLVING_FINISHED	Rješavanje provjere je završeno.
TST_AUTO_EVALUATION_PENDING	Čeka se automatsko vrednovanje.
TST_AUTO_EVALUATION_IN_PROGRESS	Automatsko vrednovanje je u tijeku.
TST_MANUAL_EVALUATION_PENDING	Čeka se ručno vrednovanje.
TST_MANUAL_EVALUATION_IN_PROGRESS	Ručno vrednovanje je u tijeku.
TST_PASSED	Provjera je položena.
TST_FAILED	Provjera nije položena.
TST_CANCELED	Provjera je poništena.
TST_ERROR	Dogodila se je greška.
TST_PREPARING_IN_PROGRESS	Postupak pripreme je u tijeku.
TST_SOLVING_SUSPENDED	Rješavanje je privremeno prekinuto.

Tablica 10-2. Vrijednosti elementa TestDescriptor.autoCreationPolicy

Naziv konstante	Opis konstante
TST_ACP_ONCE	Novi primjerak provjere znanja dozvoljeno je stvoriti samo jednom.
TST_ACP_ON_FAILURE	Novi primjerak provjere znanja potrebno je stvoriti svaki puta kada je prethodni pokušaj rješavanja provjere rezultirao padom na toj provjeri.
TST_ACP_ALWAYS	Novi primjerak provjere znanja potrebno je stvarati čim se završi sa prethodnim primjerkom.

Tablica 10-3. Vrijednosti elementa ProblemInstance.gradingStatus

Naziv konstante	Opis konstante
PRINST_STAT_NOTGRADED	Zadatak još nije bodovan.
PRINST_STAT_GRADING_PENDING	Bodovanje je zatraženo, ali još nije u postupku.
PRINST_STAT_GRADING_IN_PROGRESS	Bodovanje zadatka je upravo u tijeku.
PRINST_STAT_GRADED	Bodovanje zadatka je završeno uspješno.
PRINST_STAT_GRADING_ERROR	Bodovanje zadatka je završeno neuspješno – dogodila se je neka pogreška.

Tablica 10-4. Vrijednosti elementa ProblemInstance.evalStatus

Naziv konstante	Opis konstante
PRINST_STAT_NOTEVALUATED	Zadatak još nije vrednovan.
PRINST_STAT_EVALUATION_PENDING	Vrednovanje je zatraženo, ali još nije u postupku.
PRINST_STAT_EVALUATION_IN_PROGRESS	Vrednovanje zadatka je upravo u tijeku.
PRINST_STAT_EVALUATED	Vrednovanje zadatka je završeno uspješno.
PRINST_STAT_EVALUATION_ERROR	Bodovanje zadatka je završeno neuspješno – dogodila se je neka pogreška.

Tablica 10-5. Vrijednosti elementa ProblemInstance.problemInstantiationStatus

Naziv konstante	Opis konstante
PRINST_INST_NOT_INSTANTIATED	Primjerak zadatka još nije stvoren.
PRINST_INST_INSTANTIATING	Primjerak zadatka je u stvaranju.
PRINST_INST_INSTANTIATED	Primjerak zadatka je uspješno stvoren.
PRINST_INST_INSTANTIATION_ERROR	Primjerak zadatka nije uspješno stvoren – dogodila se je pogreška.

Tablica 10-6. Metode pomoćnog objekta ITestStartCheckerSupport

Metode za manipulaciju provjerom znanja
boolean addStartRequest(TestInstance testInstance, String queueName, String userData); Metoda u zadani red dodaje zahtjev za davanje dozvole za omogućavanje početka pisanja zadanog primjerka provjere znanja.
boolean isStartRequested(TestInstance testInstance, String queueName); Metoda provjerava postoji li u zadanom redu zahtjev za davanje dozvole za omogućavanje početka pisanja zadanog primjerka provjere znanja.
Test getTestForThisUser(String testDescriptorId); Metoda dohvaća provjere znanja za trenutnog korisnika koje pripadaju zadanom opisniku testa. Postojanje ove metode je važno jer se time sprječava pristup svim provjerama svih korisnika, i omogućava definiranje složene politike kolegija na temelju rezultata drugih provjera znanja istog korisnika.
Pomoćne metode za dohvaćanje ponuđenih usluga i upravljanje podatkovnim sjednicama
Object getBean(String beanName); Metoda dohvaća objekt koji nudi odgovarajuće usluge (objekt se navodi putem imena).
void releaseSession() throws DataStorageException; Metoda za otpuštanje svih veza sa podatkovnim sjednicama koje korisnik (možda) drži nad trenutnim primjerkom provjere znanja. U slučaju bilo kakve pogreške metoda generira pogrešku pod nazivom DataStorageException.
void rejoinSession() throws DataStorageException; Metoda za ponovno uspostavljanje svih veza sa podatkovnim sjednicama koje korisnik treba kako bi pristupio trenutnom primjerku provjere znanja. U slučaju bilo kakve pogreške metoda generira pogrešku pod nazivom DataStorageException.
TestInstance getTestInstance() throws DataStorageException; Metoda dohvaća trenutni primjerak provjere znanja. U slučaju bilo kakve pogreške metoda generira pogrešku pod nazivom DataStorageException.
void updateTestInstance() throws DataStorageException; Metoda osigurava da će sve promjene izvršene nad trenutnom provjerom znanja biti vidljive svima. U slučaju bilo kakve pogreške metoda generira pogrešku pod nazivom DataStorageException.

Tablica 10-7. Metode pomoćnog objekta ITestSupervisorSupport

Pomoćne metode za dohvaćanje ponuđenih usluga i upravljanje podatkovnim sjednicama
void releaseSession() throws DataStorageException; Metoda za otpuštanje svih veza sa podatkovnim sjednicama koje korisnik (možda) drži nad trenutnim primjerkom provjere znanja. U slučaju bilo kakve pogreške metoda generira pogrešku pod nazivom DataStorageException.
void rejoinSession() throws DataStorageException; Metoda za ponovno uspostavljanje svih veza sa podatkovnim sjednicama koje korisnik treba kako bi pristupio trenutnom primjerku provjere znanja. U slučaju bilo kakve pogreške metoda generira pogrešku pod nazivom DataStorageException.
TestInstance getTestInstance() throws DataStorageException; Metoda dohvaća trenutni primjerak provjere znanja. U slučaju bilo kakve pogreške metoda generira pogrešku pod nazivom DataStorageException.
void updateTestInstance() throws DataStorageException; Metoda osigurava da će sve promjene izvršene nad trenutnom provjerom znanja biti vidljive svima. U slučaju bilo kakve pogreške metoda generira pogrešku pod nazivom DataStorageException.

Tablica 10-8. Statusi koje vraća metoda TestController.onAction

Naziv konstante	Opis konstante
TC_A_CONTINUE	Definira da se proces provjere znanja nastavlja.
TC_A_FINISH	Definira da se proces provjere znanja završava.
TC_A_FREEZE	Definira da se proces provjere znanja privremeno suspendira.

Tablica 10-9. Metode koje mora implementirati svaka komponenta IRepositories

`Iterator<IRepository> listRepositories();` Metoda koja vraća popis svih dostupnih repozitorija. Iterator se pri tome definira kao objekt koji ima dvije metode: `boolean hasNext();` koja provjerava ima li još nedohvaćenih elemenata te `Object next();` koji dohvaća sljedeći element.
`IRepository getRepository(String name);` Metoda koja na temelju imena dohvaća odgovarajući repozitorij.
`void putRepository(String name, IRepository repository);` Metoda koja pod zadanim imenom u kolekciju dodaje predani repozitorij.
`int size();` Metoda koja vraća informaciju koliko je sveukupno repozitorija prisutno.

Tablica 10-10. Metode koje mora implementirati svaka komponenta IRepository

`String getName();` Metoda vraća ime repozitorija.
`IKeyRepository getKeyRepository();` Metoda vraća pridružen repozitorij jednostavnih vrijednosti.
`IAttachmentRepository getAttachmentRepository();` Metoda vraća pridružen repozitorij privitaka.
`void setName(String name);` Metoda postavlja ime repozitorija.
`void setKeyRepository(IKeyRepository keyRepository);` Metoda pridružuje repozitorij jednostavnih vrijednosti.

`void setAttachmentRepository(`
`  IAttachmentRepository attachmentRepository);`
Metoda pridružuje repozitorij privitaka.

Tablica 10-11. Metode koje mora implementirati svaka komponenta IKeyRepository

`byte[] getKey(String name);` Metoda vraća vrijednost pohranjenu pod zadanim nazivom.
`Iterator<String> listKeys();` Metoda vraća popis svih naziva koji imaju pohranjenu vrijednost u repozitoriju.
`void setKey(String name, byte[] value);` Metoda pod zadanim nazivom pohranjuje predanu vrijednost. Budući da se kao vrijednost predaje polje okteta, vrijednosti mogu biti proizvoljni sadržaji (brojevi, znakovni nizovi i sl.).
`int size();` Metoda vraća broj pohranjenih vrijednosti.
`void deleteKey(String name);` Metoda briše vrijednost pohranjenu pod zadanim imenom.
`boolean keyExists(String name);` Metoda provjerava postoji li vrijednost pohranjena pod zadanim imenom.

Tablica 10-12. Metode koje mora implementirati IAttachmentRepository

`Iterator<IAttachment> listAttachments();` Metoda vraća popis privitaka pohranjenih u ovom repozitoriju.
`IAttachment getAttachment(String name);` Metoda dohvaća privitak čije ime odgovara zadanom.
`void putAttachment(String name, IAttachment attachment);` Metoda pod zadanim imenom pohranjuje predani privitak.
`int size();` Metoda vraća broj pohranjenih privitaka.

Tablica 10-13. Metode koje mora implementirati svaka komponenta IAttachment

`String getName();` Metoda dohvaća ime privitka.
`String getMime();` Metoda dohvaća mime-tip privitka.
`byte[] getData();` Metoda dohvaća sadržaj privitka.
`void setName(String name);` Metoda postavlja ime privitka.
`void setMime(String mime);` Metoda postavlja mime-tip privitka.
`void setData(byte[] data);` Metoda postavlja sadržaj privitka.

10.2 Primjeri razvijenih prleta

U nastavku su prikazani primjeri najvećeg dijela dinamičkih zadataka koji su razvijeni za potrebe kolegija Digitalna logika, bez posebnih komentara. Pri tome su zadaci preneseni direktno iz Web preglednika. Zadaci su razvrstani po tipovima, kako slijedi:

- ABC pitalica s jednim točnim odgovorom
- Unos teksta
- Unos liste
- Zadaci složenog korisničkog sučelja

Za dinamičkim ABC pitalicama s više točnih odgovora nije se pojavila potreba pa konkretni zadaci nisu niti rađeni. Svi ovdje prikazani zadaci pripadaju grupi dinamičkih zadataka, kod kojih se parametri zadatka odabiru posredstvom slučajnog mehanizma u trenutku stvaranja primjerka zadatka. Svi zadaci koji nisu ABC pitalice također su opremljeni vrednovateljima, koji znaju kako se taj zadatak rješava, i koji provjeravaju je li korisnik unio ispravno rješenje. Najveći dio zadataka također je opremljen generatorom točnog rješenje, koji će u slučaju unosa netočnog rješenja za studenta stvoriti točno rješenje, koje će mu potom biti prikazano. Za ABC pitalice sam sustav nudi implementaciju vrednovatelja.

Tip zadatka: ABC pitalica s jednim točnim odgovorom

Točno Relativni doprinos: 1.0/1.0

21 podatkovni bit potrebno je zaštititi pomoću Hammingovog koda. Koliko pri tome iznosi redundancija kodiranja?

- 0.125
- 0.192
- 0.16
- 0.25

Točno Relativni doprinos: 1.0/1.0

Zadana je Booleova funkcija f(A, B, C) = ((C OR ((A OR NOT A) AND NOT NOT B)) OR (C AND A)) . Koja je od sljedećih njena komplementarna funkcija?

- ((C AND NOT (NOT B AND B)) AND A)
- (A AND (NOT B AND NOT NOT (C OR C)))
- ((NOT C AND ((NOT A AND NOT NOT A) OR NOT NOT NOT B)) AND (NOT C OR NOT A))
- (C AND (NOT A AND A))

Točno Relativni doprinos: 1.0/1.0

Kodna riječ 01011100000111111010 dobivena je zaštitom podatka 0101110000011111101 paritetnim bitom. Odredite upotrebljavanu vrstu pariteta.

- neparan
- paran

Točno Relativni doprinos: 1.0/1.0

Podatak 1010100000 potrebno je zaštititi paritetnim bitom uz uporabu neparnog pariteta. Koja je vrijednost paritetnog bita?

- 0
- 1

Točno Relativni doprinos: 1.0/1.0

Paritetnim bitom potrebno je zaštititi 15 bitova. Kolika je redundancija ovog kodiranja?

- 0.094
- (•) 0.063
- 0.115
- 0.075

Točno Relativni doprinos: 1.0/1.0

20 podatkovnih bitova potrebno je zaštititi pomoću Hammingovog koda. Koliko je potrebno zaštitnih bitova?

- 6
- 3
- (•) 5
- 7

Točno Relativni doprinos: 1.0/1.0

Hammingovim kodom potrebno je zaštititi podatkovnu riječ 10011110001010001011100, korištenjem neparnog pariteta. Točno zaštićena riječ je:

- 101100111110001101000101111100
- (•) 001000111110001101000101111100
- 011000101110001101000101111100
- 111100101110001101000101111100

Točno Relativni doprinos: 1.0/1.0

Zadane su dvije kodne riječi nekog koda, riječ
A=00001101110010011000000 i riječ
B=10010011101101110001010. Koliko iznosi njihova
distanca?

- 8
- 10
- 12
- 14 ✓

Točno Relativni doprinos: 1.0/1.0

Poprečnim i uzdužnim parnim paritetom potrebno je zaštiti
256 bitova podataka. Koliko u optimalnom slučaju iznosi
redundancija kodiranja?

- 0.0398
- 0.1142 ✓
- 0.0239
- 0.0358

Točno Relativni doprinos: 1.0/1.0

Poznato je da je minimalna distanca nekog kodiranja jednaka
21. Koliko najviše pogrešaka je moguće otkriti?

- 20 ✓
- 16
- 27
- 19

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Digitalni sustav radi s naponom napajanja od 7V i na frekvenciji od 181MHz. Uz pretpostavku da se napon napajanja moze mijenjati (uz ispravan rad sustava), kako treba promijeniti napon ako se zeli podići
frekvencija rada sklopa za 14% pri čemu potrošnja (tj. dinamička disipacija snage) treba ostati ista ?

- Napon treba povećati na 7.07[V].
- Napon treba smanjiti na 6.74[V].
- Napon treba smanjiti na 6.56[V]. ✓
- Napon treba smanjiti na 6.83[V].

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Stroj s konačnim brojem stanja zadan je tablicom u nastavku. Stroj ima jedan 1-bitni ulaz, te jedan 1-bitni izlaz.

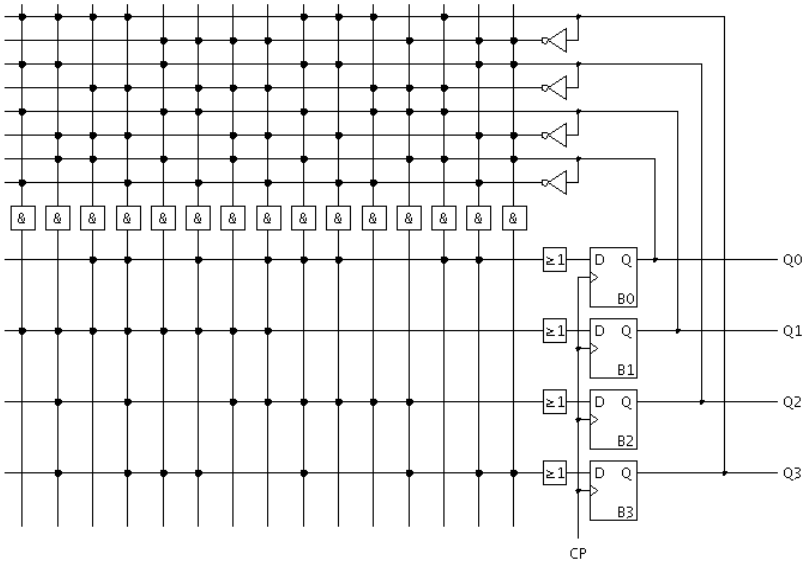
Trenutno stanje	Pobuda U	Sljedeće stanje	Izlaz
S0	0	S3	0
S0	1	S0	1
S3	0	S0	0
S3	1	S1	0
S2	0	S0	1
S2	1	S3	0
S1	0	S0	0
S1	1	S0	1

Provjerite je li zadani stroj s konačnim brojem stanja Mealyjev ili Mooreov? Dobro razmislite o vašem odgovoru!

- Stroj s konačnim brojem stanja je Mooreov, ali mu izlaz ovisi i o trenutnim ulazima
- (•) Stroj s konačnim brojem stanja je Mealyjev
- Stroj s konačnim brojem stanja je Mooreov
- Stroj s konačnim brojem stanja je Mealyjev, ali mu izlaz ne ovisi o trenutnim ulazima

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Neko sinkrono brojilo prikazano je sljedećom slikom.



U kojem ciklusu broji to brojilo? Prilikom očitavanja stanja izlaz Q0 tretirati kao bit najmanje težine.

- (14, 8, 7, 15, 11, 3, 5, 13, 2, 0, 12, 10, 6, 4, 1, 9, 14)
- (4, 7, 5, 14, 3, 12, 0, 9, 2, 13, 10, 6, 8, 15, 11, 1, 4)
- (•) (7, 12, 3, 0, 4, 10, 13, 8, 2, 5, 14, 1, 9, 6, 11, 15, 7)
- (9, 3, 12, 5, 8, 15, 13, 14, 2, 11, 1, 6, 0, 7, 10, 4, 9)

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Zadan je sklop koji je ostvaren kao sinkrono brojilo s 4 T bistabila. Poznato je da stanje 1 pripada glavnom ciklusu brojila (prilikom očitavanja stanja izlaz Q0 bistabila B0 promatrati kao bit najmanje težine).
Sljedeće stanje svakog bistabila je opisano ovim sumama minterma:
T3(Q3,Q2,Q1,Q0) = m(0,2,3,6,9,11,12,15),
T2(Q3,Q2,Q1,Q0) = m(0,2,9,10,12,14),
T1(Q3,Q2,Q1,Q0) = m(3,4,5,6,7,11,12,13,15),
T0(Q3,Q2,Q1,Q0) = m(1,6,7,10,12,13,14).
Ima li zadani sklop siguran start?

- ○ da, sklop ima siguran start, zato što ispravan rad sklopa ovisi o početnom stanju
- ○ ovisi; ovisno o početnom stanju sklop može imati siguran start
- ○ da, sklop ima siguran start
- ◉ ne, sklop nema siguran start

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Izlaz Y nekog sklopa definiran je izrazom Y <= (A OR B OR C) AND (NOT A OR B OR C) AND (NOT A OR NOT B OR C); Koju će vrijednost poprimiti taj izlaz ako se kao pobuda dovede A='U', B='0', C='1'?

- ○ Nema dovoljno informacija da bi se odgovorilo na pitanje.
- ○ '0'
- ◉ '1'
- ○ 'U'

Tip zadatka: Unos teksta

Točno Relativni doprinos: 1.0/1.0

Broj X kodiran Grayevim kodom glasi 1100100. Kako izgleda taj isti broj kodiran binarnim kodom?

1000111

Napomena: obratite pažnju da broj bitova u obje riječi mora biti jednak.

Točno Relativni doprinos: 1.0/1.0

Koji je rezultat oduzimanja brojeva: 68460 i 59888 u bazi 10 ? Rješenje mora imati isti broj znamenaka kao i zadani brojevi te biti u obliku B komplementa (rješenja s predznakom poput -101 se neće priznavati). Bilo koje redundantno proširivanje ili skraćivanje rezultata povlači netočnost zadatka.

08572

Točno Relativni doprinos: 1.0/1.0

Definirana je funkcija f(A, B, C, D). Kako izgleda algebarski zapis njenog minterma 5? Rješenje unesite u obliku npr. a and b and not c. Unos oblika f=a and b and not c je pogrešan!!!

not a and b and not c and

Točno Relativni doprinos: 1.0/1.0

Pronaći minimalni oblik funkcije f(A,B,C,D)=M(0,1,2,4,6,12) u obliku sume parcijalnih produkata. Rješenje unesite u obliku npr. a and not b and c or a and b and not c. Unos oblika f = a and not b and c or a and b and not c je pogrešan!!!!

(a and c) or (b and d) or (c

Točno Relativni doprinos: 1.0/1.0

Broj 56 u dekadskoj bazi pretvori u heksadekadsku.

38

Točno Relativni doprinos: 1.0/1.0

Neka Booleova funkcija f zadana je tablično:

A	B	C	D	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Zapišite ovu funkciju kao produkt maksterma, npr. : M1*M5 ili M(1,5).

M(4,6,7,8,10,11,13)

Točno Relativni doprinos: 1.0/1.0

Koji je rezultat izračuna 8 komplementa broja: 5735626 u bazi 8 ? Rješenje mora imati isti broj znamenaka kao i zadani broj. Bilo koje redundantno proširivanje ili skraćivanje rezultata povlači netočnost zadatka.

2042152

Točno Relativni doprinos: 1.0/1.0

Broj 100101 u binarnoj bazi pretvori u dekadsku bazu.

37

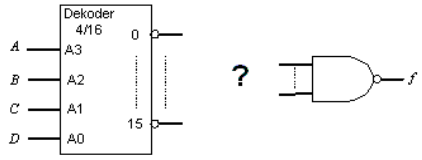
Točno Relativni doprinos: 1.0/1.0

Funkciju (A AND ((NOT NOT A AND C) OR B)) prikažite korištenjem samo NI operatora. Za prikaz koristite prefiks notaciju (npr. funkciju A OR B treba prikazati kao: NAND(NAND(A,A),NAND(B,B))).

NAND(NAND(A,NAND(N

Točno Relativni doprinos: 1.0/1.0

Na raspolaganju je dekoder 4/16 s nisko aktivnim izlazima, prikazan slikom. Ostvarite funkciju f(A,B,C,D)=(((NOT D OR D) AND D) OR A). Odredite sve izlaze koje treba spojiti na logički sklop ni, te ih unesite odijeljene zarezima.



f(A,B,C,D)= 1,3,5,7,8,9,10,11,12,13,14

Važna napomena: U polje za unos treba unijeti samo numeričke vrijednosti indeksa izlaza dekodera koje je potrebno povezati na ulaze logičkog sklopa da bi se ostvarila zadana funkcija. npr: 0,1,8,15,16,22,23,26,27 .

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Za neki digitalni sklop poznati su sljedeći podaci: UOLmax=1.06V, UOHmin=4.01V, UILmax=1.79V, UIHmin=3.06V. Sklop radi s naponom napajanja od 5V. Koliko iznosi granica istosmjerne smetnje ovog sklopa (Ugs)?

0.73 V

Važna napomena: potrebno je unijeti apsolutnu vrijednost iznosa granice smetnje, bez mjerne jedinice. Podrazumijevana mjerna jedinica je Volt.

Točno Relativni doprinos: 1.0/1.0

Neku funkciju f od 7 varijabli potrebno je realizirati uporabom multipleksora 32/1. Funkcija se realizira tako da se na adresne ulaze multipleksora dovedu varijable najvećih težina (odgovarajućim redosljedom). Pri takvoj realizaciji na podatkovne ulaze multipleksora dovode se rezidualne funkcije. Odredite broj varijabli potreban za realizaciju rezidualnih funkcija. Odgovor upišite kao broj: npr. "2" (bez navodnika).

2

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Ulazno kodersko polje obavlja potpuno dekodiranje adrese. Određuje broj riječi memorije. Od kojih logičkih sklopova se polje sastoji? Rješenje je potrebno unijeti kao ime sklopa (I, ILI, NE, NI, NILI).

I

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Za realizaciju binarnog asinkronog brojila koje broji u skraćenom ciklusu duljine 43 stanja na raspolaganju su padajućim bridom okidani T bistabili s asinkronim ulazom za postavljanje koji su svi spojeni zajedno. Pri tome je utrošen minimalno potreban broj bistabila. Stanje 0 treba biti sastavni dio ciklusa. Koje stanje treba dekodirati kako bi se realiziralo to brojilo? Kao rješenje unesite broj stanja u dekadskom sustavu (npr. 12).

42

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

U VHDL-u je definiran sklop čije je sučelje:

```
ENTITY sklop IS PORT (
   e: IN std_logic ;
   g: IN std_logic ;
   j: IN std_logic ;
   t: OUT std_logic ;
   x: OUT std_logic )
END sklop;
```

Osim toga, sklop također ima definirane i sljedeće interne signale:

```
   o: std_logic;
   q: std_logic;
```

Za taj sklop kroz sučelje aplikacije WebISE napravljen je ispitni sklop pod nazivom sklop_tb. Napišite SVE signale koje simulator simulira i koje ćete vidjeti prilikom simulacije, odnosno prilikom obavljanja ispitivanja. Napomena: ovo uključuje sve signale dizajna.

```
/sklop_tb/e
/sklop_tb/g
/sklop_tb/j
/sklop_tb/t
/sklop_tb/x
/sklop_tb/uut/e
/sklop_tb/uut/g
/sklop_tb/uut/j
/sklop_tb/uut/o
/sklop_tb/uut/q
/sklop_tb/uut/t
```

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

VHDL-om je opisan sklop čije sučelje sadrži tri ulaza (A, B i C) te jedan izlaz (f). Definirana su i tri interna signala (f1, f2 i f3). Arhitektura sklopa sastoji se od jednog bloka PROCESS, kako je prikazano u nastavku.

```
process (a, b, c)
begin
  f1 <= (NOT A AND NOT B AND C) OR (A AND NOT B AND NOT C) OR (A AND NOT B AND C) OR (A AND B AND C);
  f2 <= (NOT A AND NOT B AND NOT C) OR (NOT A AND NOT B AND C) OR (NOT A AND B AND C) OR (A AND B AND C);
  f3 <= (NOT A AND NOT B AND NOT C) OR (NOT A AND NOT B AND C) OR (NOT A AND B AND NOT C) OR (A AND B AND C);
  f  <= (NOT f1 AND NOT f2 AND NOT f3) OR (NOT f1 AND NOT f2 AND f3) OR (NOT f1 AND f2 AND f3) OR (f1 AND NOT f2 AND f3);
end process;
```

Rad ovog sklopa provjerava se tako da se na njegove ulaze dovode sve kombinacije, pri čemu u trenutku t=0ns ABC poprimaju vrijednost 000, u t=100ns 001, u t=200ns 010, ... te u t=700ns 111, čime završava pobuda. Izračunajte kakav će točno biti rezultat simulacije na izlazu f (prikažite si to i grafički). U kutiju za unos rješenja potrebno je unijeti rezultat simulacije izlaza f. Rješenje se sastoji od slijeda uređenih parova (vrijeme, vrijednost). Primjerice, ako ste dobili da funkcija f u trenutku t=0ns poprima vrijednost 1, pa u t=300ns poprima vrijednost 0, pa već u sljedećoj delti se vraća na 1, te u t=500ns pada na nulu, kao rješenje ćete upisati (0,1)(300,0)(300,1)(500,0), bez ikakvih razmaka.

(100,1)(200,0)(300,1)(400,

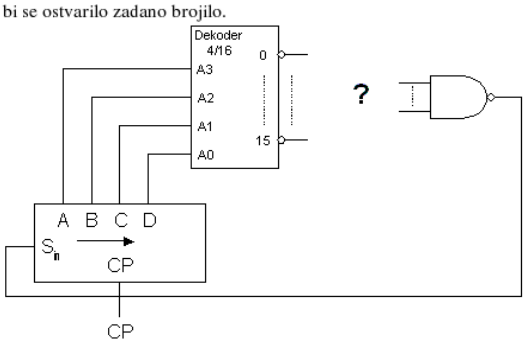
Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Za realizaciju binarnog asinkronog brojila koje broji u skraćenom ciklusu duljine 30 stanja na raspolaganju su padajućim bridom okidani T bistabili s asinkronim ulazom za brisanje koji djeluju kada im se dovede logička jedinica (svi su spojeni zajedno; označimo tu točku oznakom X). Brojilo treba ostvariti minimalno potrebnim brojem bistabila, pri čemu stanje 0 treba pripadati ciklusu. Pobudu za točku X generira kombinacijski sklop. Koju funkciju taj sklop treba ostvarivati? Kao rješenje upišite algebarski oblik (npr. not Q2 or Q1). Prilikom očitavanja stanja izlaz Q0 smatra se izlazom najmanje težine.

Q4 AND Q3 AND Q2 ANI

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Potrebno je projektirati sklop koji će prolaziti kroz sljedeća stanja: (4, 10, 13, 6, 11, 5, 2, 1, 8, 12, 14, 7, 3, 9, 4). Sklop je potrebno ostvariti uporabom strukture prikazane na slici (posmačni registar + dekoder 4/16 s nisko aktivnim izlazima). Nespecificirana stanja treba tako riješiti da sklop u minimalnom broju koraka dođe u ciklus (bilo gdje)! Ukoliko se može u minimalnom broju koraka vratiti u ciklus u više stanja, treba odabrati stanje niže vrijednosti. Koje sve izlaze treba spojiti na logički sklop ni ? U polje za unos odgovora je potrebno unijeti indekse izlaza dekodera (vidi sliku), odijeljene zarezima, koje je potrebno povezati na ulaze logičkog sklopa da bi se ostvarilo zadano brojilo.



izlazi dekodera 0, 1, 3, 4, 6, 8, 10, 12

Važna napomena: U polje za unos treba unijeti samo numeričke vrijednosti indeksa izlaza dekodera koje je potrebno povezati na ulaze logičkog sklopa da bi se ostvarilo zadano brojilo. npr: 0,1,8,15,16,22,23,26,27 .

Tip zadatka: Unos liste

Točno

Zadana je funkcija f(A,B,C,D)=m(0,6,12)+d(1,5,7,9,10,11,14,15). Potrebno je minimizirati zadanu funkciju metodom Quine-McCluskey, te odrediti sve primarne implikante, sve bitne primarne implikante te sve minimalne zapise funkcije u obliku sume parcijalnih produkata.

Važna napomena: ukoliko u neko polje treba unijeti više rješenja (primjerice ako funkcija ima više primarnih implikanata), tada je potrebno svako rješenje unijeti u zaseban redak.

Primarni implikanti	`not a and not b and not c`
	`a and b and not d`
	`b and c`

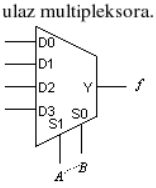
Bitni primarni implikanti	`not a and not b and not c`
	`a and b and not d`
	`b and c`

Minimalni zapisi	`not a and not b and not c or a and b and not d or b and c`

Važna napomena: ova pomoć je dohvaćena iz repozitorija problema, kako se ne bi duplicirala u svim instancama.

Točno Relativni doprinos: 1.0/1.0

Na raspolaganju je multipleksor 4/1 prikazan slikom. Ostvarite funkciju f(A,B,C)=(A OR ((A OR NOT A) OR NOT B)). Selekcijski ulazi u multipleksor su A,B. Što treba dovesti na podatkovne ulaze multipleksora? U slobodna polja potrebno je unijeti algebarski oblik funkcije koja određuje pojedini ulaz multipleksora.



D0 1

D1 1

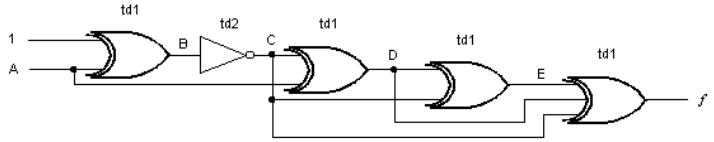
D2 1

D3 1

Važna napomena: U svako polje za unos treba unijeti samo jedan algebarski oblik funkcije koja se dovodi na pojedini ulaz multipleksora. Ulaze treba realizirati bez varijabli dovedenih na selekcijske ulaze, te ukoliko se to ne poštuje, zadatak će biti ocijenjen kao netočan. Dodatna napomena: za konstantne vrijednosti su dozvoljena dva načina unosa vrijednosti: 1 se tretira jednako kao i true, a 0 je ekvivalentna sa false.

Djelomično točno Vaše rješenje | Točno rješenje Relativni doprinos: 0.4/1.0

Zadan je sklop prema slici. Pretpostaviti da su svi signali početno u stanju 0. Kašnjenje sklopova td1 iznosi 108 ns, a kašnjenje sklopova td2 iznosi 54 ns. Nacrtajte dijagram koji prikazuje ovisnost svih signala o vremenu, i s tog dijagrama očitajte vrijednosti označenih signala u trenutku t=125 ns. Poznato je da u trenutku t=0 ns ulaz A poprima vrijednost 1. Tražene vrijednosti upišite u polja u nastavku.



Signal B 0

Signal C 1

Signal D

Signal E

Signal F

U svako polje za unos odgovora unijeti odgovor oblika "0" ili "1".

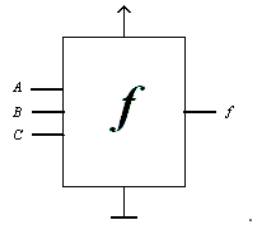
Točno Relativni doprinos: 1.0/1.0

Na raspolaganju je prioritetni koder s 8 ulaza. Izlazi prioritetnog kodera su y2,y1,y0, te z. Ako se na ulaze prioritetnog kodera dovede u7,u6,u5,u4,u3,u2,u1,u0=00101000, odredite izlaz prioritetnog kodera! Ulaz u0 je ulaz najmanje težine. Izlaz y0 je izlaz najmanje težine.

y2 1
y1 0
y0 1
z 1

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Neki digitalni sklop prikazan slikom koristi naponsku logiku, te logička stanja prikazuje naponskim razinama -3V i 5V.
Ulazi sklopa su (A, B, C), a izlaz sklopa je f. Poznato je da sklop u negativnoj logici obavlja funkciju: f=((C OR NOT (NOT C AND A)) AND (B OR A)). Odredite tablicu kombinacija napona (ovisnost izlaznog napona o naponima na ulazima sklopa). Rješenje za pojedine kombinacije unosi se u pripadni redak. Vrijednosti napona se odvajaju zarezima, prvo varijable najveće težine, pa niže, a vrijednost izlaza f za tu kombinaciju ulaznih napona dolazi zadnja. U prvi redak tablice kombinacija napona potrebno je unijeti napone za slučaj kada su svi ulazi u logičkoj nuli (sljedeći redak tablice odgovara kombinaciji napona kada je ulaz najmanje težine u logičkoj jedinici, itd).



0. redak 5,5,5,5
1. redak 5,5,-3,5
2. redak 5,-3,5,-3
3. redak 5,-3,-3,-3
4. redak -3,5,5,5
5. redak -3,5,-3,-3
6. redak -3,-3,5,5
7. redak -3,-3,-3,-3

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Na raspolaganju su TTL sklopovi podskupine P1 i P2 čiji su parametri zadani kako slijedi:
P1:I(ol)=13.31 mA, I(il)=1.92 mA, I(oh)=421.66 uA, I(ih)=40.57 uA.
P2:I(ol)=7.94 mA, I(il)=0.45 mA, I(oh)=361.23 uA, I(ih)=21.23 uA.
Sklop iz jedne podskupine pobuđuje više sklopova iz druge podskupine. Koliko se maksimalno sklopova može spojiti na izlaz jednog sklopa u oba slučaja? Oznaka P1/P2 odnosi se na slučaj kada sklop podskupine P1 pobuđuje sklopove podskupine P2.

(P1/P2) Izlaz je L: 29

(P1/P2) Izlaz je H: 19

(P1/P2) Zajednički uvjet: 19

(P2/P1) Izlaz je L: 4

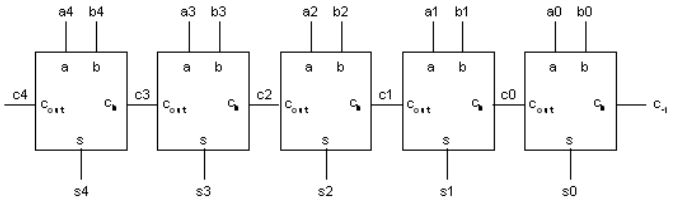
(P2/P1) Izlaz je H: 8

(P2/P1) Zajednički uvjet: 4

Važna napomena: U polje za unos treba unijeti samo broj sklopova (cijeli broj)! npr. 4.

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Na slici je prikazano paralelno binarno zbrajalo. Ako svako potpuno zbrajalo kasni 10ns, a u trenutku t=0ns na C_{-1} se dovede 1 te a4a3a2a1a0 = 11101, b4b3b2b1b0 = 00011, nacrtajte vremenske dijagrame svih signala, te sa njih očitajte vrijednosti svih izlaza S i C u trenutku t = 33ns. Prilikom rješavanja zadatka pretpostaviti da su vrijednosti svih izlaza (rezultat i prijenos) u trenutku t = 0ns jednaki nula!



S(0)= 1
C(0)= 1
S(1)= 0
C(1)= 1
S(2)= 0
C(2)= 1
S(3)= 1
C(3)= 0
S(4)= 1
C(4)= 0

Važna napomena: U svako polje za unos treba unijeti samo vrijednost pripadnog izlaza rezultata i prijenosa zbrajala. Dozvoljena su dva načina upisa vrijednosti: 1 se tretira jednako kao i true, a 0 je ekvivalentna sa false.

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Potrebno je realizirati sinkrono brojilo koje broji u ciklusu: (12, 6, 7, 9, 14, 8, 1, 2, 15, 5, 11, 10, 3, 0, 13, 4, 12), ako je na raspolaganju 4 T bistabila. Prilikom očitavanja stanja izlaz Q0 bistabila B0 promatrati kao bit najmanje težine. U polja za unos rješenja unijeti minimizirane algebarske oblike funkcija.

T3
T2
T1
T0

Važna napomena: U polja za unos rješenja treba unijeti logičke izraze funkcija dovedenih na ulaze bistabila, uz koje će se ostvariti zadano brojilo.

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Nacrtajte shemu 3-bitnog asinkronog binarnog brojila unaprijed izvedenog rastućim bridom okidanih JK bistabila. Za taj sklop potrebno je nacrtati vremenske dijagrame do trenutka T=1539 ns. Na brojilo se dovodi signal takta periode 171 ns, pri čemu u trenutku t=0 nastupa rastući brid. Kašnjenje svakog bistabila iznosi 38 ns. Brojilo osim bistabila ne smije koristiti dodatne logičke sklopove. Pretpostaviti da su svi bistabili u trenutku prije t=0 u stanju 0. Ako s Q0 označimo izlaz bistabila najmanje težine, očitajte stanje svih izlaza u trenutku t=1275 ns.

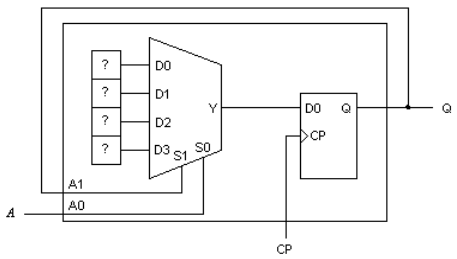
Q0 0
Q1 0
Q2 1

Važna napomena: vrijednosti koje se prihvaćaju su: 0, 1 (alternativno: true, false).

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Na raspolaganju je 2-ulazni LUT s D bistabilom, prikazan slikom. Programirati taj bistabil tako da se dobije bistabil čija je funkcija opisana sljedećom tablicom.

A	Qn+1
0	1
1	not Qn

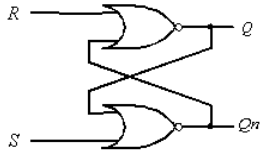


LUT_0 1
LUT_1 1
LUT_2 1
LUT_3 0

Važna napomena: U svako polje za unos treba unijeti samo vrijednost pripadnog ulaza luta za navedenu funkciju. Dozvoljena su dva načina upisa vrijednosti: 1 se tretira jednako kao i true, a 0 je ekvivalentna sa false. Bilo koji oblik redundantnog unosa (dupliciranje nula, proširivanje jedinice sa vodećom nulom) povlači netočnost unosa.

Nije riješen Vaše rješenje ! Točno rješenje Relativni doprinos: 0.0/1.0

Koje je stanje na izlazima SR bistabila realiziranog pomoću dva NILI sklopa (prema slici) u trenutku 319ns (kašnjenje td svakog NILI sklopa iznosi 10ns), ako se na ulaze bistabila dovodi pobuda prema slici? Prilikom rješavanja zadatka pretpostaviti da se svi sklopovi ponašaju idealno.

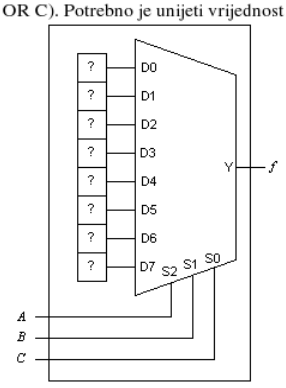


Q 1

Qn 1

Nije riješen Vaše rješenje ! Točno rješenje Relativni doprinos: 0.0/1.0

Na raspolaganju je pregledna tablica logičkog bloka sklopa FPGA (tj. LUT) sa 3 ulaza, prikazan slikom. Potrebno je realizirati funkciju: f(A,B,C)=(((NOT NOT B OR NOT NOT A) AND (NOT NOT C AND C)) OR C). Potrebno je unijeti vrijednost svakog ulaza u pripadajuće polje za unos odgovora.



D0 0

D1 1

D2 0

D3 1

D4 0

D5 1

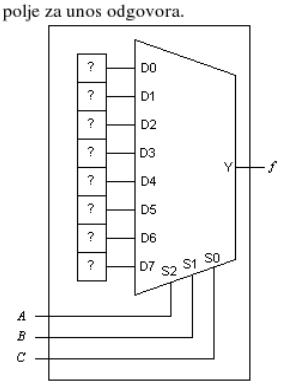
D6 0

D7 1

Važna napomena: U svako polje za unos treba unijeti samo vrijednost pripadnog ulaza luta za navedenu funkciju. Dozvoljena su dva načina upisa vrijednosti: 1 se tretira jednako kao i true, a 0 je ekvivalentna sa false.

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Na raspolaganju je pregledna tablica logičkog bloka sklopa FPGA (tj. LUT) sa 3 ulaza, prikazan slikom. Potrebno je realizirati funkciju:f(A,B,C)=M(0,3,5,7). Vrijednost svakog ulaza potrebno je unijeti u pripadajuće polje za unos odgovora.



D0
D1
D2
D3
D4
D5
D6
D7

Važna napomena: U svako polje za unos treba unijeti samo vrijednost pripadnog ulaza luta za navedenu funkciju. Dozvoljena su dva načina upisa vrijednosti: 1 se tretira jednako kao i true, a 0 je ekvivalentna sa false.

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Popunite tablicu promjene stanja T bistabila.

Q	T	Qnext
0	0	R0
0	1	R1
1	0	R2
1	1	R3

R0 0
R1 1
R2 1
R3 0

Važna napomena: vrijednosti koje se prihvaćaju su: 0, 1 (alternativno: true, false). Ukoliko je neka pobuda za bistabil zabranjena, tada za sljedeće stanje pri toj pobudi unesite znak minus (-) ili slovo x.

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Realizirati sinkrono brojilo koje broji u sljedećem ciklusu: (14, 11, 8, 6, 10, 2, 1, 12, 15, 13, 9, 14), ako su na raspolaganju 4 T bistabila. Osigurati siguran start brojila. Prilikom očitavanja stanja brojila izlaz Q0 bistabila B0 promatrati kao bit najmanje težine. U polja za unos odgovora potrebno je unijeti algebarski zapis funkcije ulaza tih bistabila.

Prilikom unosa algebarskog oblika za stanja bistabila koristiti oznake Qj (gdje je j broj bistabila; npr. Q2).

Primjer jednog takvog rješenja:

Q2 and not Q1 or not Q0.

B3.T

B2.T

B1.T

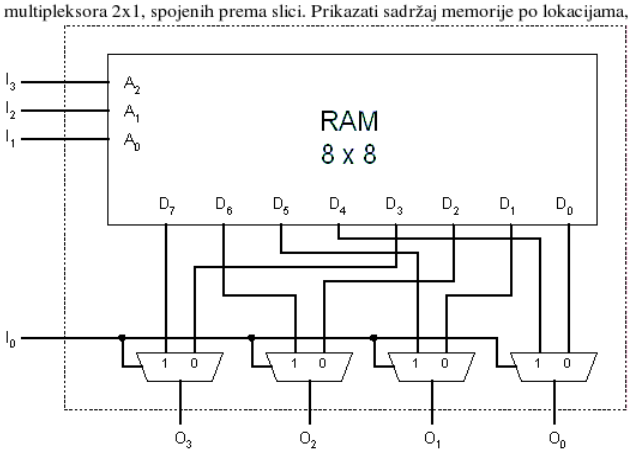
B0.T

Napomene vezane uz ocjenjivanje zadatka

Ovaj zadatak privremeno nije ocijenjen.

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Zadana je uređena n-torka P=(0, 14, 12, 8, 2, 10, 1, 4, 8, 3, 13, 5, 10, 13, 4, 3). Funkcija F(i) vraća i-ti element od P (npr. F(6) = 1). Projektirati sklop koji ostvaruje ovu funkciju. Na raspolaganju je ispisna memorija 8x8 te 4 multipleksora 2x1, spojenih prema slici. Prikazati sadržaj memorije po lokacijama, u heksadekadskom obliku.



Lokacija 0: E0

Lokacija 1: 8C

Lokacija 2: A2

Lokacija 3: 41

Lokacija 4: 38

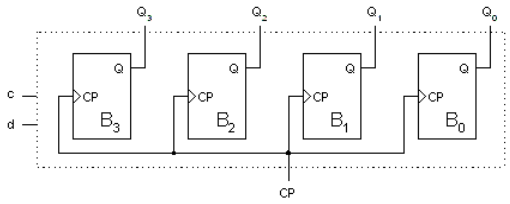
Lokacija 5: 5D

Lokacija 6: DA

Lokacija 7: 34

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

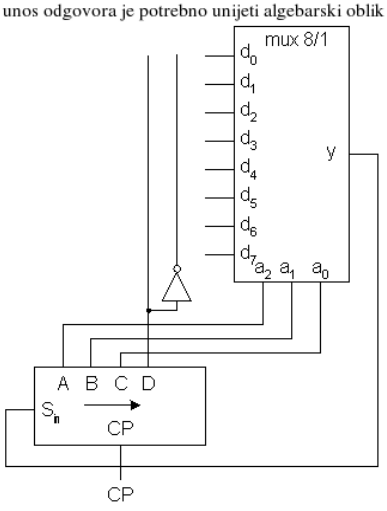
Uporabom T bistabila realizirati 4-bitno brojilo koji broji ovisno o signalu d: ako je d=1, tad je sljedeće_stanje = trenutno_stanje-3, inače sljedeće_stanje = trenutno_stanje+1 (pod pojmom stanje misli se na binarno kodirani broj zapisan kroz bistabile, pri čemu je izlaz Q0 izlaz najmanje težine). Sklop treba imati i sinkroni ulaz za brisanje c (kojeg bistabili nemaju). Koristiti minimalni broj osnovnih logičkih sklopova. U svako polje za unos potrebno je unijeti algebarski zapis funkcije tog bistabila. Prilikom očitavanja stanja izlaz Q0 bistabila B0 tretirati kao izlazni bit najmanje težine.
Prilikom unosa algebarskog oblika za stanja bistabila koristiti oznake Qj (gdje je j broj bistabila; npr. Q2).
Primjer jednog takvog rješenja:
c and Q2 and not Q1 or not c and d and not Q0.



B3.T (NOT c AND NOT d AND
B2.T (NOT c AND NOT d AND
B1.T (NOT c AND Q0) OR (c A
B0.T (NOT c) OR (Q0)

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Potrebno je projektirati sklop koji će prolaziti kroz sljedeća stanja: (8, 4, 10, 13, 6, 11, 5, 2, 9, 12, 14, 7, 3, 1, 8). Sklop je potrebno ostvariti uporabom strukture prikazane na slici (posmačni registar + multipleksor). Nespecificirana stanja treba tako riješiti da sklop u minimalnom broju koraka dođe u ciklus (bilo gdje)! Ukoliko se može u minimalnom broju koraka vratiti u ciklus u više stanja, treba odabrati stanje niže vrijednosti.
Selekcijski ulazi u multipleksor su: A, B, C. Što treba dovesti na podatkovne ulaze multipleksora? U polja za unos odgovora je potrebno unijeti algebarski oblik funkcije kojom je određen pojedini ulaz multipleksora.



D0 1
D1 NOT D
D2 NOT D
D3 NOT D
D4 D
D5 NOT D
D6 NOT D
D7 0

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Uporabom JK bistabila i minimalnog broja dodatnih logičkih sklopova ostvariti D bistabil. Rješenje, za svaki ulaz zasebno, mora biti u minimalnom obliku.

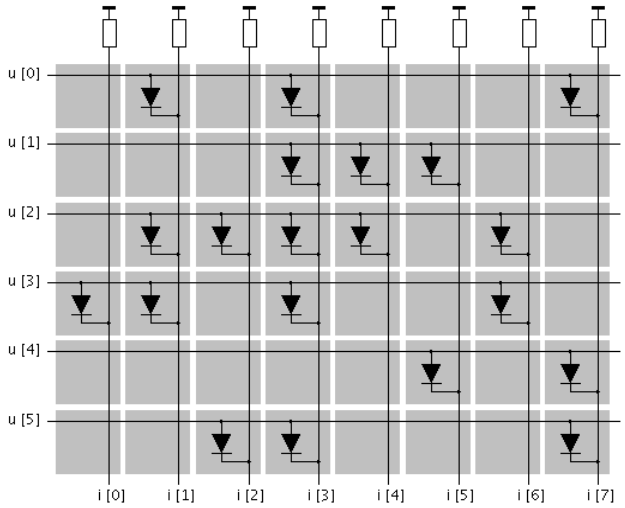
J (D)

K (NOT D)

Oznaka za trenutno stanje je Q, a za sljedeće Qn

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Permanentna memorija je zadana slikom.



Očitajte sadržaj memorije po lokacijama. U polja za unos rješenja potrebno je unijeti vrijednost memorijske lokacije u heksadekadskom obliku (kao dvije heksadekadske znamenke); npr. E8 ili 2F. Pri tome bit i[0] tretirajte kao bit najveće težine.

Lokacija 0: 51

Lokacija 1: 1C

Lokacija 2: 7A

Lokacija 3: D2

Lokacija 4: 05

Lokacija 5: 31

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

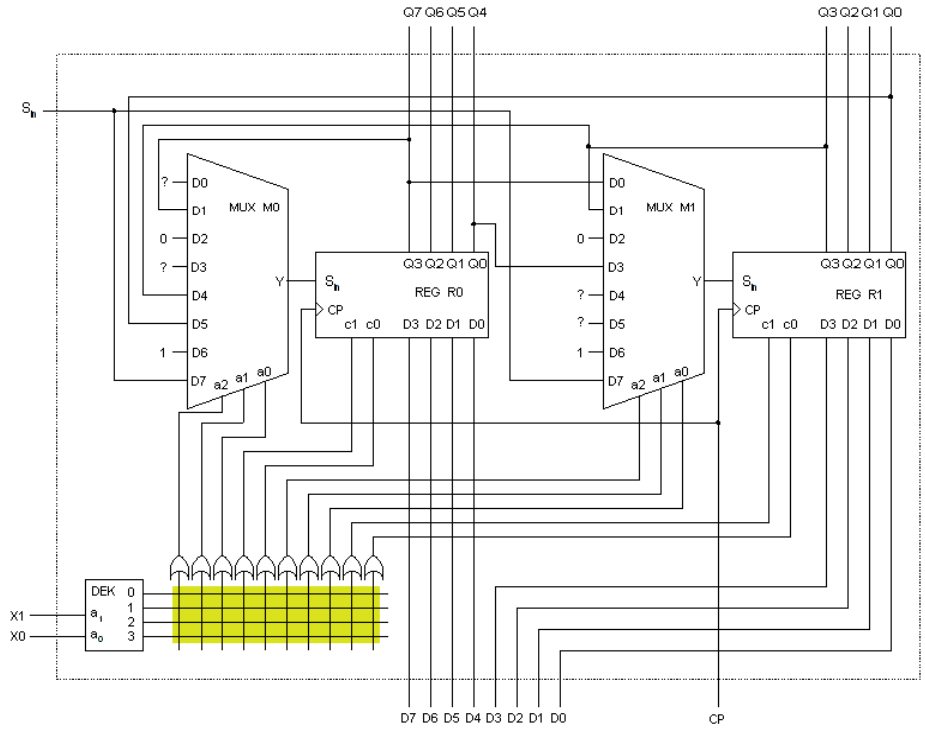
Funkcija 4 bitnih registara prikazanih na slici definirana je sljedećom tablicom.

C1	C0	Opis
0	0	Posmak u lijevo uz čuvanje bita predznaka, punjenje sa Sin
0	1	NOP - nema nikakve promjene
1	0	Paralelni upis
1	1	Posmak u lijevo, punjenje sa Sin

Uporabom multipleksora i dekodera s programirljivom ILI ravninom ostvarena je struktura koja se može programirati tako da se dobije 8-bitni registar s mnoštvom podržanih operacija. Vaš je zadatak programirati ovu strukturu tako da se dobije jedan 8-bitni registar čija je funkcija određena sljedećom tablicom.

X1	X0	Opis
0	0	Ciklički posmak u lijevo (rotacija)
0	1	Paralelni upis
1	0	Posmak u lijevo uz čuvanje bita predznaka, punjenje sa Sin
1	1	Posmak u lijevo, punjenje sa Sin

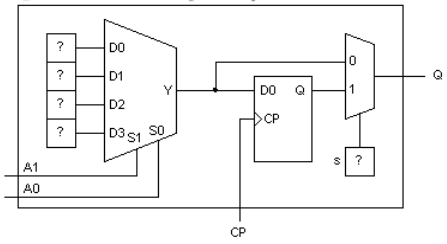
Ulaze u multipleksor označene upitnikom zabranjeno je koristiti. U polja za unos rješenja u nastavku potrebno je unijeti zarezima odvojen popis izlaza dekodera koje je potrebno spojiti na odgovarajući ILI sklop, kako bi se ostvarila potrebna funkcija. Ako se za neki ILI sklop ne definira niti jedan izlaz dekodera, taj ILI sklop na izlazu generira logičku nulu.



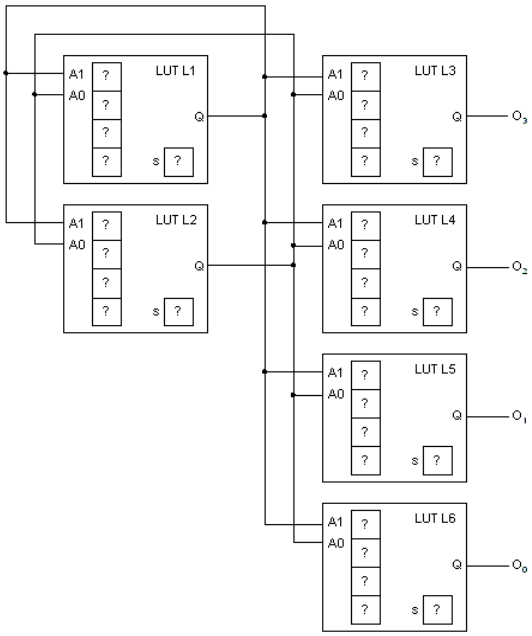
M0.a2	0,2,3
M0.a1	
M0.a0	
R0.c1	0,1,3
R0.c0	0,3
M1.a2	2,3
M1.a1	2,3
M1.a0	2,3
R1.c1	0,1,2,3
R1.c0	0,2,3

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Logički blok FPGA sklopa temeljen na LUT-u, bistabilu i multipleksoru prikazan je na sljedećoj slici.



Više takvih logičkih blokova povezano je u sklop prikazan u nastavku.



Programirajte sve logičke blokove tako da se dobije sklop koji na izlazu generira sekvencu:
2, 2, 5, 1.
Pri tome izlaz O3 tretirati kao izlaz najveće, a izlaz O0 kao izlaz najmanje težine.

L1.lut	0,1,1,0
L1.s	1
L2.lut	1,0,1,0
L2.s	1
L3.lut	0,0,0,0
L3.s	0
L4.lut	0,0,1,0
L4.s	0
L5.lut	1,1,0,0
L5.s	0
L6.lut	0,0,1,1
L6.s	0

Važna napomena: U polja za unos u koje treba unijeti vrijednost pregledne tablice (Li.lut) unosi se popis zarezima odvojenih vrijednosti. Pri tome se najprije unosi vrijednost koja odgovara nultoj lokaciji pregledne tablice. U polja za unos u koje treba unijeti vrijednost adresnog ulaza multipleksora (Li.s) unosi se jedan broj.

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Stroj s konačnim brojem stanja zadan je tablicom u nastavku. Stroj ima jedan jednobitni ulaz, te jedan 3-bitni izlaz.

Trenutno stanje	Pobuda U	Sljedeće stanje	Izlaz
S2	0	S3	5
S2	1	S3	6
S1	0	S1	1
S1	1	S2	4
S0	0	S3	0
S0	1	S0	7
S3	0	S1	6
S3	1	S3	0

Za realizaciju tog stroja na raspolaganju su 2 SR bistabila. Pri tome se svako stanje kodira prema tablici prikazanoj u nastavku.

Stanje	Kod stanja	
	Q1	Q0
S0	1	0
S1	0	1
S2	0	0
S3	1	1

Projektirajte taj sklop uporabom zadanih bistabila. Nije dozvoljeno obavljati minimizaciju broja stanja stroja (naime, iako se ovo uobičajeno čini prilikom projektiranja, za potrebe strojnog ocjenjivanja u ovom se zadatku ne smije). U polja za unos rješenja za svaki ulaz bistabila, te za svaki izlaz stroja unesite minimizirani algebarski zapis funkcije.

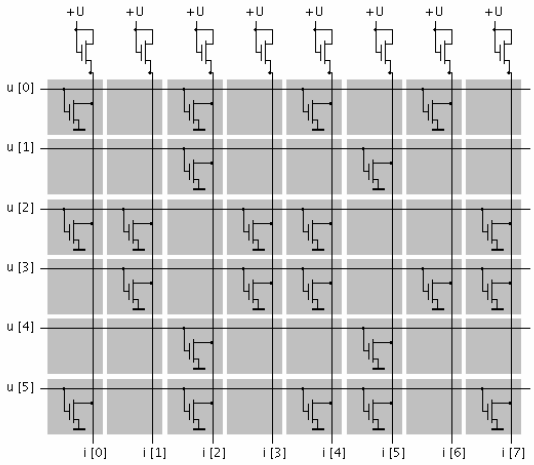
Prilikom unosa algebarskog oblika za stanja bistabila koristiti oznake Qj (gdje je j broj bistabila; npr. Q2), odnosno U za pobudu. Primjer jednog takvog rješenja:

Q2 and not Q1 and U or not Q0 and not U. Prilikom očitavanja izlaza stroja bit izlaza O0 promatran je kao bit najmanje težine.

B1.S	(NOT Q0)
B1.R	(Q0 AND NOT U)
B0.S	(NOT Q1 AND NOT Q0) (
B0.R	(NOT Q1 AND Q0 AND U
O2	(NOT Q1 AND NOT Q0) (
O1	(Q1 AND Q0 AND NOT U
O0	(NOT Q1 AND NOT U) O

Nije riješen Vaše rješenje | Točno rješenje Relativni doprinos: 0.0/1.0

Permanentna memorija je zadana slikom.



Očitajte sadržaj memorije po lokacijama. U polja za unos rješenja potrebno je unijeti vrijednost memorijske lokacije u heksadekadskom obliku (kao dvije heksadekadske znamenke); npr. E8 ili 2F. Pri tome bit i[0] tretirajte kao bit najveće težine.

Lokacija 0: 55

Lokacija 1: DB

Lokacija 2: 26

Lokacija 3: A4

Lokacija 4: DB

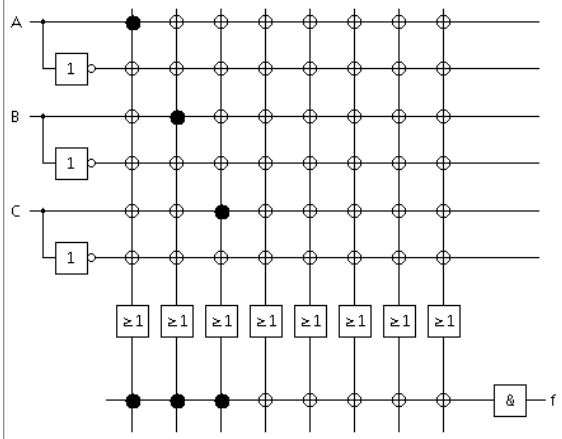
Lokacija 5: 52

Tip zadatka: Zadaci složenog korisničkog sučelja

Točno

Relativni doprinos: 1.0/1.0

Zadana je funkcija f=((A AND B) AND ((B OR C) AND C)). Potrebno je klikanjem na spojeve isprogramirati zadani PLA sklop tako da izvršava zadanu funkciju.

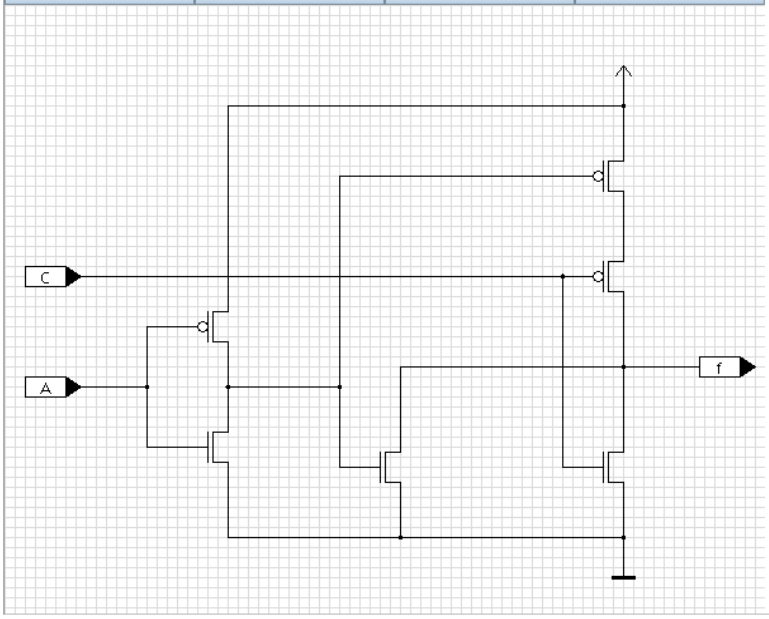


Nije riješen Vaše rješenje | Točno rješenje

Relativni doprinos: 0.0/1.0

Zadana je funkcija f([A, B, C])=(NOT (C OR NOT A) AND A). Realizirajte ovu funkciju CMOS tehnologijom.
Napomena: prilikom rješavanja zadatka možete se poslužiti simulatorom CMOS sklopova dostupnim na Webu. U simulatoru najprije nacrtajte vaš sklop i provjerite radi li ispravno (kako biste izbjegli probleme poput loših spojeva i sl.). Tek kad ste sigurni da sklop radi ispravno, snimite ga u clipboard, i učitajte u zadatak u zadaći.

Obriši označeno | Obriši sve | Učitaj | Snimi



Točno Relativni doprinos: 1.0/1.0

Zadana je funkcija f(A,B,C,D)=m(6,7,10,11,14)+d(2,3,4,9,12,15). Potrebno je minimizirati zadanu funkciju metodom Quine-McCluskey, te odrediti sve primarne implikante, bitne primarne implikante te sve minimalne zapise funkcije.

Primarni implikanti | Bitni primarni implikanti | Minimalni zapisi

a and not b and d
b and not d
c

Uputa: Potrebno je u svaki redak upisati po jedan primarni implikant funkcije.

Životopis

Rođen sam u Zagrebu 09. lipnja 1979. godine, gdje sam s odličnim uspjehom završio osnovnu i srednju školu. Diplomirao sam s odličnim uspjehom na Fakultetu elektrotehnike i računarstva 26. rujna 2002. na smjeru Računarstvo, s diplomskim radom pod nazivom “Inteligentno pretraživanje informacijskog prostora informacijske infrastrukture”, i stekao titulu diplomiranog inženjera računarstva. Nagrađen sam nagradom "Josip Lončar" za najboljeg studenta na smjeru 2001. godine. Također sam primio nagradu "Stanko Turk" za osobito vrijedan diplomski rad iz polja računarstva, kao i "Brončanu plaketu Josip Lončar" za postignuti uspjeh tijekom studija. Dobio sam Rektorovu nagradu kao koautor za rad "Web-orijentirani simulator modela neuronskih mreža s primjerom raspoznavanja novčanica i generatorom uzoraka".

Sukladno odobrenju Ministarstva znanosti i tehnologije zaposlen sam na Fakultetu elektrotehnike i računarstva na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave od 07. siječnja 2003. kao znanstveni novak na projektu 0036033 – Semantički Web kao omogućavatelj informacijske infrastrukture.

Član sam nekoliko svjetski priznatih strukovnih udruga: ACM – Association for Computing Machinery, IEEE – Institute of Electrical and Electronics Engineers te AAAI - American Association for Artificial Intelligence.

Naslov

Model sustava e-ispitivanja s potporom za inteligentno upravljanje provjerom znanja

Sažetak

U radu je dan kratak uvod u sustave e-učenja, i trenutno važeće norme na ovom području. Potom je ukratko analizirana potpora e-ispitivanju poznatih sustava WebCT i MOODLE, kao i prethodno razvijenog sustava WODLS. Dan je i kratak osvrt na sustav Nescume, koji se koristi za pristup sustavu e-ispitivanja StudTest.

Uveden je model sustava StudTest, i definirani su osnovni koncepti tog modela. Ukratko je opisana uloga definiranih koncepata i njihov međusobni odnos. Definiran je i način interakcije između pojedinih implementacija koncepata odgovarajućih modela.

Razrađena je mogućnost uporabe definiranih koncepata za izvedbu sustava koji omogućava inteligentno (ili prilagodljivo) vođenje ispitnog procesa. Ilustrirano je što treba napraviti i na koji način, kako bi se pružila ovakva usluga.

Opisana je prototipna implementacija sustava e-ispitivanja zasnovana na modelu StudTest. Opisana je i korištena tehnologija te razlozi njene uporabe, kao i konkretna implementacija koncepata samog modela.

Potom su ilustrirane mogućnosti razvijenog sustava kroz niz razvijenih zadataka, uz kratke komentare o njihovim mogućnostima, načinu postavljanja pitanja i načinu vrednovanja studentovih odgovora.

Iznesen je kritički osvrt na razvijeni model, njegove karakteristike, način primjene sustava u stvarnoj nastavi te postignuti rezultati.

Rad sadrži i dodatak s većim dijelom zadataka razvijenih za potrebe kolegija Digitalna logika.

Title

Model of e-examination system with support for intelligent assessment management

Abstract

In this work a short introduction to e-learning systems and current relevant standards is given. E-examination support of recognized systems WebCT and MOODLE is analyzed, as well as of previously developed system WODLS. A short description of Nescume system, which is used for accessing e-examination system StudTest, is presented.

StudTest model and relevant concepts, such as *prlets*, are then defined. A short description of defined concepts is given, as well as their relationship. Interaction among defined concepts is also defined.

A way for using defined concepts for intelligent (or adaptive) assessment management is described. Then, it is illustrated what must be done, in order to provide this service.

A prototype implementation of e-examination system is described, based on StudTest model. Also, used technology and reasons for its usage is elaborated, as well as the implementation of model concepts.

Then, capabilities of developed system are illustrated, through a series of developed *prlets*, along with comments on their capabilities, on the way of asking questions and on techniques of answer correctness evaluation.

A critical review of developed model, its characteristics, usage in education process and results are given.

This work also contains an appendix, with most of *prlets* developed for course Digital logic.

Ključne riječi

e-ispitivanje, e-učenje, inteligentno vođenje ispitnog procesa

Keywords

e-examination, e-learning, intelligent assessment management