Goran Šimić,
major
ŠC veze, informatike, EI i PED,
Beograd

WEB UČIONICA

UDC: 681.324 : 007.52° : 378.147

Rezime:

U ovom radu opisana je arhitektura, dizajn i implementacija Code Tutor-a, inteligentnog tutorskog sistema zasnovanog na Web-u, koji je namenjen obuci studenata u oblasti radio-komunikacija. Code Tutor je dizajniran kao Web učionica i klijent-server sistem, izgraden korišćenjem savremenih inteligentnih i Internet tehnologija. Iskustva sa Code Tutor-om pokazuju da i nastavnici i studenti imaju pozitivna mišljenja vezana za korišćenje sistema kao podrške za učenje različitih domenskih tema. U cilju ilustrovanja praktične primene Code Tutor-a, materijal predstavlja neke detalje studentske i nastavničke sesije sa sistemom.

Ključne reči: inteligentni tutorski sistem, Web tehnologije, arhitekture, dizajn, aplikacija.

WEB CLASSROOM

Summary:

This paper describes architecture, design, and implementation of Code Tutor, a Web-based intelligent tutoring system that facilitates learning of radio communications to the students of a telecommunications college. Code Tutor, designed as a Web classroom client-server system, is ontologically founded, and built using modern intelligent and Web-related technologies. Experience with Code Tutor so far shows that both teachers and learners have positive feelings about using Code Tutor as a support tool for learning different topics in the domain. In order to illustrate the use of Code Tutor in practice, the paper also presents some details of both students' and teachers' sessions with the system.

Key words: intelligent tutoring system, Web technologies, architectures, design, application.

Uvod

Dostupnost novih Internet tehnologija otvorila je mnoge mogućnosti za informatičku podršku u procesu nastave. Na serverskom (nastavničkom) računaru moguće je instalirati neki od Web servera (Apache, Tomcat ili neki drugi) i koristiti ga za smeštanje nastavnih HTML stranica. Na klijentskim računarima (studentski) mogu se pregledati HTML stranice (statičke ili dinamički kreirane). Zahtevi na klijentskoj strani su minimalni: potreban je bilo koji Web pretraživač (Internet Explorer, Netscape Navigator, Opera ili neki drugi). Primer jednog takvog sistema je Code Tutor.

Code Tutor je mali tutorski sistem zasnovan na Internet tehnologiji. Dizajniran je za brzu pripremu i proveru stude-

nata za praktične vežbe u oblasti radiokomunikacija. Pošto se završi teorijski deo predmeta, studenti se praktično obučavaju u korišćenju skupe opreme. U toku izvođenja vežbi nastavnik odgovara za ispravnost opreme i njeno pravilno korišćenje. Ova dva zahteva su neretko protivrečna (u slučaju nedovoljno pripremljenih studenata). Malo je vremena da nastavnik stekne lični uvid i proceni individualnu pripremljenost svakog studenta. Da bi se navedeni problemi rešili razvijen je sistem Code Tutor.

Celokupan sistem implementiran je u programskom jeziku Java. Za razvoj je korišćen JBuilder 6, alat za razvoj Java i Web aplikacija (Borland, 2001). Sistemski srednji (poslovni) sloj dizajniran je korišćenjem Rational Rose alata za softversku analizu i dizajn (Rational, 2000). Novi Code Tutor integrisao je i druge tehnologije:

– CLIPS, alat za razvoj ekspertnih sistema (CLIPS, 2002), korišćen je za generisanje fajlova sa bazom znanja;

– ekspertski sistem shell zasnovan na Java-i, Jess je korišćen za interpretiranje tih fajlova (Jess, 2002), (Friedman-Hill, 2002);

– za komunikaciju učenika sa sistemom korišćen je standardni Web pretraživač;

– Java™ Servlet tehnologija (Sun, 2001), (Hall, 2001a), (Hall, 2001b), korišćena je za implementaciju interakcije učenika sa sistemom;

– Apache server (Apache, 2001a), korišćen je za smeštanje statičkih HTML stranica;

– Apache JServ (Apache, 2001b), korišćen je za interpretiranje servleta;

– XML tehnologija (W3C, 2001), korišćena je u generisanju fajlova koje Code Tutor koristi za obezbeđivanje preporuka učenicima tokom korišćenja sistema.

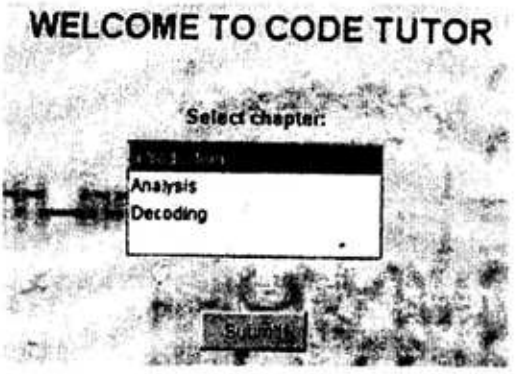
Materijal demonstrira mogućnosti korišćenja pomenutih tehnologija u izvođenju nastave. Sledeći odeljak analizira sistemske slučajeve korišćenja i sistemsku arhitekturu. Detalji dizajna i implementacije predstavljeni su u trećem odeljku, dok četvrti odeljak upućuje na inteligentno ponašanje Code Tutora. Poslednji odeljak sadrži važne detalje korisničkog interfejsa.

Sistemska analiza i arhitektura

U sistemu postoje dva glavna aktera: student (na klijentskoj strani) i nastavnik (na serverskoj strani). Radi toga postoje dva generalna skupa slučaja korišćenja: klijentski i serverski.

SLUČAJEVI KORIŠĆENJA KLIJENTSKE STRANE

U sistemu postoje četiri modela studentske interakcije sa sistemom Code Tutor, i svaki od njih predstavlja poseban slučaj korišćenja. Interakcija studenta sa sistemom započinje autentikacijom *(Autentication)* – procesom logovanja za novu sesiju učenja. Učenje *(Learning)* počinje izborom jedne od ponuđenih tema *(Chapters)*. Svaka tema sadrži jednu ili više lekcija *(Lessons)*, koje predstavljaju osnovnu nastavnu jedinicu. Teme su predstavljene HTML stranicama nastavnog materijala, a lekcije paragrafima u HTML stranici teme. Student proučava

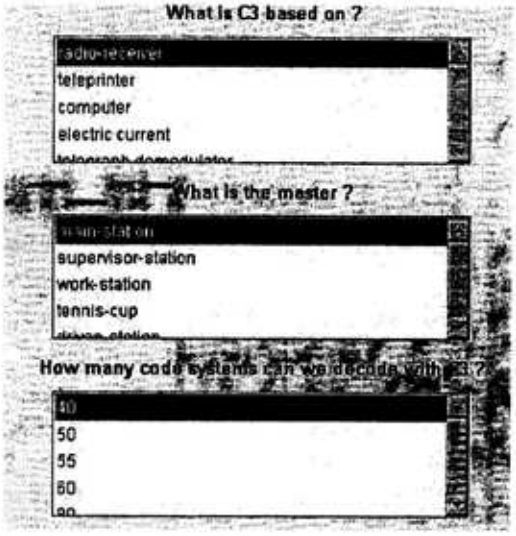


Sl. 1 – Fragment HTML stranice koju generiše „Start" servlet

tekstualne stranice, koje su obogaćene multimedijalnim sadržajima, kao što su ilustracije vremenskih i spektralnih dijagrama i audio klipovi.

Izučavanje lekcija jedne teme završava se proverom znanja. Pri testiranju *(Assessment)* studenti odgovaraju na pitanja. Pošto unesu sve odgovore, oni potvrđuju korisnički unos. Ta potvrda (klik na komandno dugme) aktivira ocenjivanje *(Validation)*. Da bi ocenio studenta, sistem izvršava niz funkcionalnih poziva po dubini modela s ciljem proveravanja i ažuriranja modela studenta, kao i s ciljem ocenjivanja njegovih znanja vezanih za analizu i prijem različitih tipova radioemisija. Sistem daje ocenu za svaki odgovor i izračunava prosečnu ocenu. Pojedinačne ocene agregiraju u konačnu ocenu domenskog znanja studenta. Ako učenik na testu dobije jednu ili više slabih ocena, sistem ga vraća na početak odgovarajuće lekcije.

Student koji završi testiranje sa pozitivnom ocenom ima dva izbora: da odabere drugu temu, ili da ponovi istu temu da bi popravio lični uspeh. U prvom slučaju sistemski procesi su isti, kao što su opisani u prethodnom paragrafu. U



Sl. 2 – HTML stranica koju generiše „Test" servlet

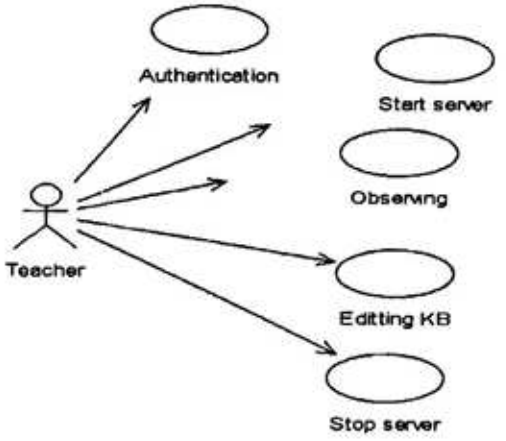
slučaju ponavljanja, Code Tutor pomaže studentu kroz preporuke *„Šta bi bilo dobro ponoviti za bolji uspeh"*.

SLUČAJEVI KORIŠĆENJA SERVERSKE STRANE

Nastavnik je akter na serverskoj strani. Njegovi zadaci su autentikacija, pokretanje servera, praćenje sesija studenata, editovanje baze znanja i zaustavljanje servera (slika 3).

Neki od zadataka znatno su različiti u odnosu na klijentsku (studentsku) stranu, kao što je, na primer, editovanje baze znanja. Za razliku od korisničkog interfejsa klijentske strane, na kojoj je samo pretraživač, nastavnička strana sadrži veći broj opcija.

Autentikacija nastavnika slična je studentskoj *(Authentication)*, ali sa drugačijim svojstvima. Nastavnik može da pokrene server, bez čega je nemoguće korišćenje sistema Code Tutor. Podrazu-



Sl. 3 – Slučajevi korišćenja serverske strane aplikacije

meva se da se server pokreće uvek nakon nastavničke autentikacije.

Kad se učenik prijavi (uloguje) u sistem, serverska strana aplikacije kreira novu instancu korisnika (nadalje klasa *User*). Svaka instanca smešta studentov identifikacioni broj i druge podatke korisnika koji su relevantni za sesiju. Te instance koriste objekat klase *Sesija* (izveden iz Java klase *HttpSession*). Kada student odabere temu za izučavanje, njen naziv se, takođe, smešta u objekat sesije. Nakon što tutor oceni odgovore studenata, rezultat se smešta u odgovarajuću instancu klase *User*. Objekti koji su korišćeni u sesiji su serijabilni, kako bi nastavnik mogao da pristupi podacima studenata i nakon završetka sesije *(Observing)*.

Jedan od najznačajnijih modula na serverskoj strani aplikacije jeste editor baze znanja. Nastavnik može dodavati, menjati i brisati lekcije. Na primer, on može odabrati temu i jednu od lekcija, a zatim editovati deo koji želi. Kada je lekcija modifikovana, nastavnik potvrđuje promene. Kao glavni efekat ove operacije, Code Tutor generiše skript koji ažurira fajl baze znanja (CLIPS fajlovi za baze znanja imaju ekstenziju *.clp*).

Svi studenti moraju završiti svoje sesije pre nego što nastavnik zaustavi Web server *(Stop server)*, što je neophodno da bi sistem održao podatke studenata konzistentnim.

ARHITEKTURA SISTEMA CODE TUTOR

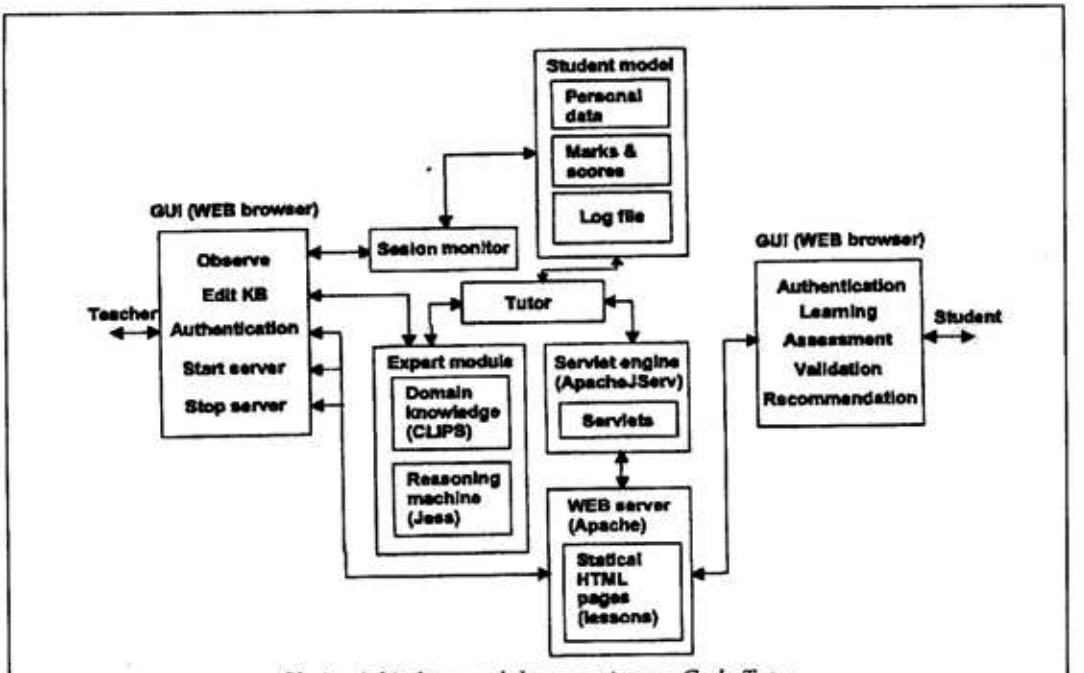
Na osnovu analize slučajeva korišćenja proizišla je arhitektura sistema Code Tutor koja je predstavljena na slici 4. Centralizovana arhitektura izražena je kroz jedinstveni repozitorijum studentskih modela i istovremenu podršku za više studenata.

Dizajn i implementacija

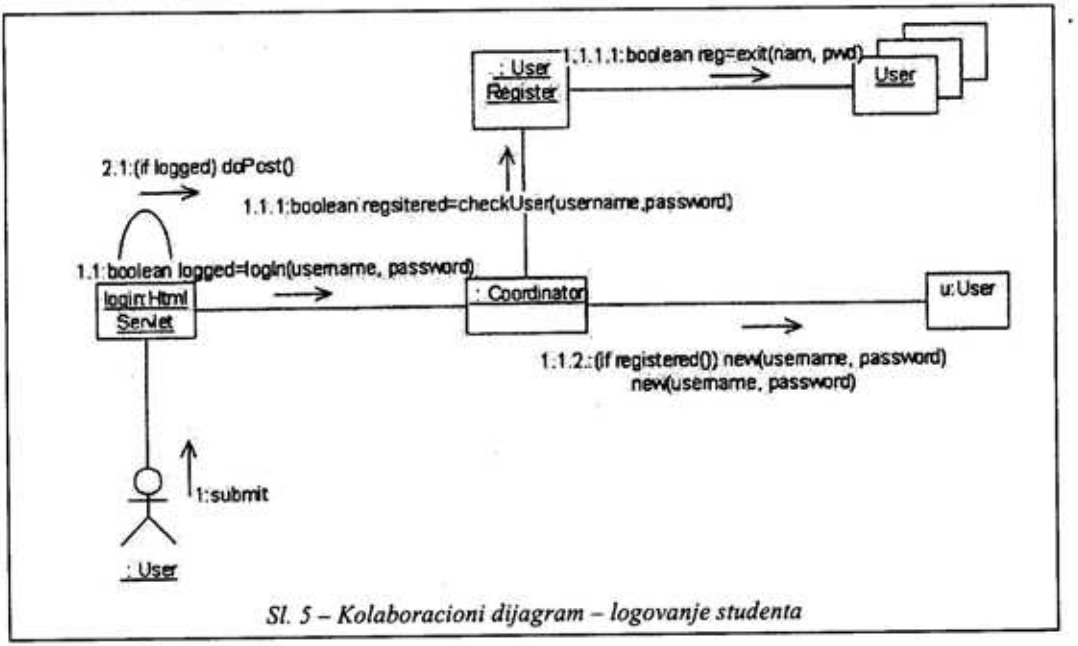
Na osnovu slučajeva korišćenja i funkcionalnih opisa i analize modelovano je ponašanje sistema kroz sistemske sekvencijalne i kolaboracione dijagrame.

STUDENTSKA SESIJA
Logovanje

Nakon što student pokrene pretraživač, unosi URL prvog servleta. Kao rezultat, generiše se HTML stranica sa kontrolama za unos imena i lozinke za logovanje korisnika. Tokom inicijalizacije, servlet kreira instancu *Koordinator* klase (slika 5) i priključuje je u objekat *Sesije*. Koordinator je dizajniran po GRASP paternu – controller (Larman,



Sl. 4 – Arhitektura celokupnog sistema Code Tutor



Sl. 5 – Kolaboracioni dijagram – logovanje studenta

1999). Inicijalizacija se dalje prostire po dubini modela, a koordinator kreira objekat klase *Registar Korisnika*. Ovaj objekat se puni podacima studenata iz eksternog fajla, pri čemu *Koordinator* proverava autoritet studenta. Ako je registrovan za korišćenje sistema, *Koordinator* kreira novu instancu klase *User* i smešta je u

heš tabelu (Java klasa *Hashtable*). Tada *Koordinator* šalje poruku servletu da se može pozvati jedna od metoda sledećeg servleta (*doPost* ili *doGet*).

Učenje

Prvi servlet za učenje je i poslednji. Kada student odabere temu za izučavanje, *Koordinator* pamti izbor i koristi ga kasnije kao jedan od parametara za testiranje. Studenti mogu krstariti HTML stranicama i učiti željenu temu. Sistem servleta referencira statičke HTML stranice sa mnogobrojnim multimedijalnim sadržajima. U ovoj fazi uključen je samo servlet za učenje.

Provera

Za inicijalizaciju testiranja sistem poziva „Test" servlet. Ime teme je argument prosleđen „Test" servletu (kroz objekat *Sesije*). *Koordinator* mapira naziv teme na putanju do baze znanja (*.clp fajla), pokreće Jessovu mašinu za zaključivanje (objekat klase *Rete*) i rezultat je punjenje radne memorije sadržajem referenciranog *clp* fajla. To su pravila od kojih neka bivaju aktivirana odmah po unošenju – pravila koja sistemu šalju pitanja i odgovore za testiranje. Druga pravila se aktiviraju tek pošto studenti završe test.

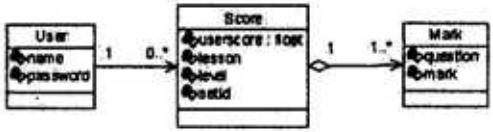
Ocenjivanje

Student kroz testiranje čita pitanja, selektuje odgovore za koje smatra da su najtačniji, a na kraju potvrđuje unos. Tada se odgovori smeštaju automatski u objekat *Sesije* i poziva se servlet za ocenjivanje. U ovom servletu *Koordinator* ocenjuje korisnika korišćenjem metode *validateAnswers*. Kod ove metode odgovori se šalju u Jess mašinu za zaključivanje. Vrši se određivanje ocena u sistemu radnih pravila i na kraju se izračunava prosečna ocena. Servlet prikazuje te rezultate kroz HTML stranicu, ali i smešta u korisnički *Rezultat* objekat koji je serijabilan. Ako student ima jednu ili više negativnih ocena, sesija se nastavlja sa servletom za učenje (ponavljanje istih HTML stranica – stranica teme).

NASTAVNIČKA SESIJA

Nastavnička aplikacija (na serverskoj strani) implementirana je kroz definisane forme (Frames&Dialogs). Da bi pokrenuo Web server nastavnik mora da se prijavi (uloguje) na sistem, koji najpre proverava njegov autoritet.

Pored odgovornosti za modifikovanje baze znanja, nastavnik je odgovoran i za izvršenje zadataka administracije. Jedan od njih je ažuriranje studentskih podataka. Pri radu sistema, podaci o studentima smešteni su u objekat klase Registar Korisnika. Nastavnik može pratiti rezultate studenata. U toku sesije studentski model sadrži korisničko ime, lozinku, odabranu temu (poglavlje), ocene iz testa i konačan uspeh. Sve to je agregirano u klasi Korisnik *(User)*. Principi ocenjivanja korisnika ukazuju na tri važna dela korisničke klase (slika 6).



Sl. 6 – Važni delovi studentskog modela

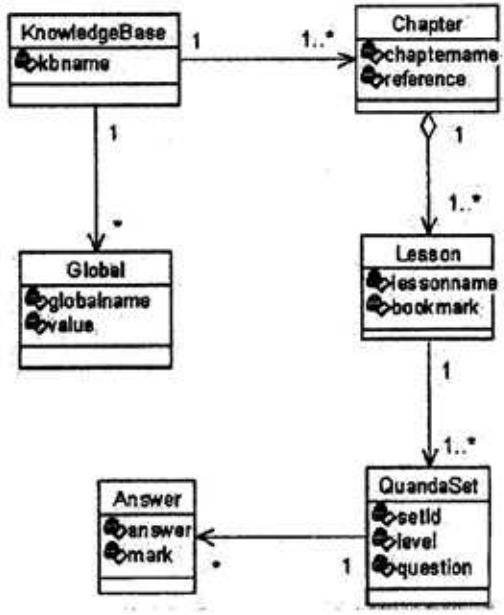
Podaci studenata su članovi u objektima koji su pridruženi klasi Korisnik *(User)*, dok je Rezultat *(Score)* agregirana iz instanci klase Ocena *(Mark)*. Funkcionalnost pojedinih članova podataka u modelu opisana je u odeljku koji obrađuje inteligentno ponašanje sistema Code Tutor.

Kada student započne novu sesiju *Koordinator* proverava autoritet korisnika. Kada se korisnik prijavi za korišćenje sistema *Koordinator* ažurira korisničke podatke u instanci korisnika. Slično se dešava i kada se student odjavljuje sa sistema. Pri odjavi *Koordinator* ažurira log fajl koji je dizajniran u XML formatu, kako bi bio dostupan nastavniku kroz Web pretraživač (korišćenjem CSS – Cascade Style Sheet formata). Ove akcije su implementirane u monitoru sesije. Svaki put kada student bude ocenjen *Koordinator* dodaje rezultate testiranja u isti fajl. Na taj način sistem formira zapis istorije učenja korisnika. Zapisi iz log fajla jedino su dostupni kroz nastavničku aplikaciju. Nastavnik može samo da čita, ali ne i da modifikuje podatke. On ne mora pokrenuti server ako želi videti rezultate studenta.

BAZA ZNANJA

Code Tutor sadrži ekspertski modul zasnovan na pravilima. Znanje je sadržano u pravilima, temama *(chapters)*, lekcijama *(lessons)* i setovima pitanja i odgovora *(quandasets)*. Znanje je struktuirano i predstavljeno ontologijom čiji su osnovni koncepti prikazani na slici 7.

Glavna klasa u ontologiji je Baza Znanja *(Knowledge Base)* i sadrži ostale koncepte ontologije znanja Code Tutora. Instance klase koja predstavlja nastavnu temu *(Chapter)* predstavljaju teme koje student može da proučava. Lekcije *(Lessons)* su delovi tema i predstavljaju nastavne jedinice. Za svaku lekciju vezan je jedan ili više setova pitanja i odgovora *(QuandaSet)*. Svaki set, u stvari, implementira pitanje sa mogućnošću višestrukog izbora (uključuje string pitanja i više odgovora). Klasa Odgovor *(Answer)* sadrži string i ocenu odgovora.



Sl. 7 – Ontologija znanja Code Tutora

Globalne varijable koriste se za smeštanje adrese delova domenskog znanja (instance tema) i broja lekcija po jednoj temi. *Global* je klasa koja sadrži samo *put* i *get* metode kao *JavaBean*. Ako nastavnik dodaje novu lekciju, odgovarajuća globalna varijabla (koja čuva broj lekcija) biva inkrementirana i obrnuto – ako nastavnik edituje postojeće lekcije globalna varijabla se ne menja.

Editovanje lekcija

Kada nastavnik edituje lekciju sistem generiše novi skript baze znanja. To znači da će pravila, koja sadrže reference za bilo koji podatak koji nastavnik može promeniti u lekciji takođe biti promenjena.

U Code Tutoru sva ontologija predstavljena je setovima pitanja i odgovora. Pošto Jess mašina za zaključivanje interpretira pravila, a ne setove pitanja i odgovora, Code Tutor automatski generiše pravila iz tih setova i pamti ih u *.clp fajl. Dok studenti (klijentska strana), koriste sistem Jess interpretira sadržaj *.clp fajla (fajla baze znanja). Kad god nastavnik dodaje, menja ili uklanja lekciju, ili ažurira domensko znanje, promene se automatski reflektuju u odgovarajućim pravilima u *.clp fajlu. Ovakav pristup omogućava nastavniku da lako menja sadržaj baze znanja, bez poznavanja programiranja na CLIPS skript jeziku. Na taj način svi delovi baze znanja dobijaju veliku koheziju, u smislu problemske fokusiranosti, i slabu spregu, u smislu uzajamne zavisnosti. Sistem stiče još jedno svojstvo: postaje skalabilan za realizaciju različitih tutorskih zadataka (implementacija različitih tehnika zaključivanja).

Inteligentno ponašanje sistema Code Tutor

Vidi se da Code Tutor funkcioniše kao proceduralni program – student bira temu iz koje odmah zatim uči posredstvom HTML stranica sa tekstovima, ilustracijama i zvukom. Nakon toga odgovara na pitanja, i konačno sistem ocenjuje njegove sposobnosti. Nadalje se nastavlja inteligentna interakcija studenta sa Code Tutorom.

SAVETI ZA UČENJE

Jedna od manifestacija inteligencije sistema jeste savet *(Recomendation)* studentu šta da uči. Na primer, nakon ocenjivanja Code Tutor može sugerisati studentu da ponovo pročita određenu lekciju. Saveti su formirani na osnovu rezultata testiranja. Postoji poseban modul baze znanja (poseban fajl) za tu namenu. Sistem preuzima rezultate testiranja (preko *User* objekta) i pretražuje ih utvrđujući najlošiju ocenu. Ako postoje dve ili više istih najslabijih ocena, sistem uzima prvu koju nađe. Na osnovu indeksa ocene dalje uzima ime lekcije na koju se ocena odnosi i nalazi mapiranu referencu na odgovarajuću HTML stranicu (paragraf). Konačan rezultat ovog procesa rezonovanja je servletski generisan link koji predstavlja referencu (URL) HTML stranice (paragrafa). Student može kliknuti mišem na preporučen link i sesija se nastavlja ponovnim čitanjem referencirane lekcije (teme).

Ukoliko je student zadovoljan ocenom, potvrđivanjem rezultata završava sesiju.

Preporuke usmeravaju studenta na odgovarajuću HTML stranicu ili, preciznije, paragraf na stranici. Mapiranje se vrši korišćenjem heš tabele (implementirane u *RefMap* klasi) koja je napunjena u toku inicijalizacije sistema. Za pamćenje referenci korišćen je XML format. Teme i lekcije sadrže reference kao atribute. Svaka tema sadrži referencu na HTML stranici (hyperlink), dok lekcije sadrže referencu na paragraf u HTML stranici (hyperlink+bookmark). Sistem čita XML fajl i puni heš tabelu, čiji je ključ naziv lekcije. Servlet koji generiše HTML stra-

nicu sa rezultatima obrađuje pojedinačne ocene i generiše referencu koja odgovara imenu lekcije iz koje je postignuta najslabija ocena. Sesija se nastavlja ako student klikne na tu referencu. Rezultati studenta su zapamćeni pre nego što Code Tutor prikaže stranicu sa rezultatima.

VIŠENIVOJSKO UČENJE

Višenivojsko učenje je druga manifestacija inteligentnog ponašanja Code Tutora. Ekspertsko znanje u sistemu Code Tutor struktuirano je u dva nivoa: osnovnom nivou i naprednom nivou. Student najpre savladava osnovne koncepte radio-komunikacija, a ako savlada osnovni test bez negativnih ocena, prelazi na savladavanje naprednog nivoa. Treba napomenuti da sistem nije limitiran na samo dva nivoa.

TEST SETOVI

Jedno od iskustava u korišćenju sistema Code Tutor je nastojanje studenata da zapamte pitanja i slabe odgovore kako bi položili test iz drugog puta. Sa takvom neadekvatnom motivacijom (učenje radi ocene a ne radi znanja), studenti brzo završavaju iščitavanje tekstova i prelaze na testiranje, jer su sigurni u uspeh. Iz toga je proisteklo da jedan set pitanja (po lekciji) nije dovoljan, tako da postoji veći broj setova pitanja i odgovora (QuandaSet instance). Studenti koji ponavljaju temu, bilo zbog negativne ocene ili želje za boljim uspehom, uvek dobijaju novi set pitanja, što je uticalo na pažljivije i ozbiljnije proučavanje lekcija. Upravo radi toga sistem je postao nešto komplikovaniji.

Uloga tutora vrlo je značajna za motivaciju studenata. Njegov cilj je da pomogne studentima u razumevanju domena i akviziciji znanja koje im treba. Ukoliko je nastavnik prestrog i ne razume potrebe studenata, motivacija opada. Code Tutor rešava taj problem tako što daje šansu studentu da ponovi test, savetujući ga šta da bolje nauči kako bi postigao bolji rezultat. Slobodno pretražujući HTML stranice student je sigurniji u sopstveni uspeh, a zahvaljujući višenivojskom učenju satisfakcija u učenju je veća.

Dizajn korisničkog interfejsa

Jedna od glavnih karakteristika u tutorskom sistemu jeste da očuva pažnju i motivaciju studenta tokom sesije učenja. Kao i kod realnih nastavnika (ljudi), više interakcije sa učenikom rezultira boljom koncentracijom i rezultatima, što zahteva višenivojski pristup. Osnovna percepcija studenta (vizuelna i audio percepcija) mora biti razmatrana isto kao i njegove kognitivne sposobnosti (čitanje, učenje, interpretiranje, primena i transfer znanja). Sva dešavanja između studenta i veštačkog tutora odvijaju se kroz korisnički interfejs.

DIZAJN I SADRŽAJ LEKCIJA

Kao što je istaknuto, teme *(Chapters)* sadrže tekstove, grafiku i audio klipove koji su dostupni posredstvom Web pretraživača (kao HTML stranice). Pozadina tih stranica dizajnirana je kao tamne, nenametljive slike koje odražavaju kontekste tema. Na slikama su spektralni i vremenski dijagrami nekih radio-emisi-

ja, ali koji ne odvraćaju studenta od učenja. Preuzeto iz prethodne verzije Code Tutora, svaka tema ima različitu sliku u pozadini. To pomaže učeniku da vizuelno zapamti gde može naći određene podatke. Pozadina, boja teksta, veličina i izgled fonta takođe uvećavaju čitljivost i jasnoću sadržaja, pomažući studentu da uči sa što manje naprezanja. U lekcijama Code Tutora korišćeno je mnoštvo ilustracija i audio-uzoraka. Svaka slika i audio-klip povezani su sa odgovarajućim paragrafom u tekstu. Ključne lekcije u nastavnim temama naznačene su i tag-ovane na detaljnija objašnjenja. Ako student nešto zaboravi, sistem mu omogućava da pretražuje niže nivoe lekcija.

Drugi aspekt je kompleksnost HTML stranice. Prenatrpanost tekstom, slikama i kontrolnim objektima (komandna dugmad i dugmad za potvrđivanje i izbor, itd.) može biti vrlo naporna za studenta. Zbog toga teme u Code Tutoru mogu biti podeljene u više HTML stranica, dok svaka ima nekoliko lekcija (koje su u suštini nastavne jedinice), tako da su pregledne i lake za snalaženje. Takođe, ontologija je raslojena – student se može uvek vratiti na stranice na nižem nivou izučavanja, ali sa nižeg nivoa može preći na viši nivo tek kada položi testove nižeg nivoa.

DIZAJN STRANICA ZA TESTIRANJE

Postoje mnoge tehnike za ocenjivanje znanja studenta. U svim testovima student ili bira odgovor od više ponuđenih (ako su zahtevani tekstualni odgovori) ili upisuje numeričke podatke u odgovarajuće polje. Da bi se zadržala jednostavnost sistema, testovi nižeg nivoa su dizajnirani tako da izvrše kategorizaciju odgovora na tačne ili netačne. Sistem registruje da li je student položio test. Tek tada on može čitati lekcije na višem nivou, kao i učestvovati u testu višeg nivoa. Testovi višeg nivoa skaliraju znanje studenta ocenom od jedan do pet. Code Tutor ne ograničava broj pitanja u testu vezanom za temu. Preporučuje se pravilo – više lekcija sa manje pitanja po lekciji. Međutim, ovo pravilo je u kontradikciji sa činjenicom da ocena bolje odslikava znanje ispitanika ako se postavi više pitanja po temi. U praksi, nastavnik će, na osnovu iskustva, doneti odluku o optimalnom broju pitanja za specificiranu temu.

Zaključak

Sadašnja verzija sistema Code Tutor, zahvaljujući korišćenju najsavremenijih tehnologija, u potpunosti podržava koncepte Web učionice. Iako je namenjena za korišćenje u realnoj učionici (intranet), sa malim dodacima vezanim za povećanje bezbednosti sistema, može se koristiti u virtuelnoj učionici. Objektivno orijentisan pristup omogućio je značajnu fleksibilnost dizajna, što je jedna od značajnijih prednosti sadašnje verzije Code Tutora u odnosu na prethodnu.

Jedan od prvih koraka u budućem razvoju sistema Code Tutor biće proširivanje njegovih mogućnosti, kako bi bio domenski nezavisan. Bez potrebe za poznavanjem ekspertskih sistema, Internet tehnologije i mrežnog programiranja, nastavnik će dizajnirati ontologiju kursa, dodavati nastavne materijale (materijali za učenje), kreirati teme, lekcije, setove

pitanja i odgovora za testiranje dok će sistem obavljati svu potrebnu automatiku: prebacivanje statičkih HTML stranica na Web server, generisanje skript-fajlova koje ekspertski sistem shell može interpretirati, generisati potrebne fajlove sa struktuiranim podacima (kao što su mape referenci) i sl. To znači da će nastavnik imati zadatak samo da markira lekcije (markiranje ključnih reči i imenovanje tema i lekcija) i smestiti ih u bazu znanja (smeštanje fajla teme na određeni direktorijum i ubacivanje reference za fajl u sistem). Nakon toga sistem treba da iskoristi nastavnički unos za punjenje XML fajla sadržajem. Takođe, sistem treba da generiše *Template* za temu i odgovarajuće forme vezane za ključne reči. Obeležena fakta u lekcijama mogu biti iskorišćena za generisanje pravila.

Nastavnik bi trebalo da razume domen i osnovne principe obeležavanja lekcija, ali nije neophodno i poznavanje detalja sistemskog dizajna.

Promene studentske strane aplikacije trebalo bi da pruže što bolje uslove za savladavanje nastavnih sadržaja. Student bi, nakon autorizacije, mogao da podešava izgled stranica (pozadinu, veličinu i vrstu slova, formu za testiranje).

Jedno od značajnijih unapređenja treba da bude povećanje inteligencije sistema, u smislu preciznije obrade istorije rezultata studenata. Korelaciona statistika ukrštanjem rezultata jednog studenta iz više kurseva, ili rezultata više nastavnih grupa za jedan kurs, pomogla bi i nastavnicima i rukovodiocima u korigovanju vođenja studenata. Pragmatičan pristup u rešavanju konkretnih problema nastavne prakse omogućio bi masovniju primenu i više ulaganja u razvoj inteligentnih tutorskih sistema.

Literatura:

[1] Hall, L. and Gordon, A.: Synergy on the Net: Integrating the Web and Intelligent Learning Environments. Proceedings of The Workshop on Web-Based ITS. San Antonio, TX, USA (electronic edition), 1998.

[2] López, J. M. et al: Design and Implementation of a Web-based Tutoring Tool for Linear Programming Problems. Proceedings of The Workshop on Web-Based ITS. San Antonio, TX, USA (electronic edition), 1998.

[3] Johnson, W. L. et al: Pedagogical Agents on the Web. Proceedings of The Workshop on Web-Based ITS. San Antonio, TX, USA (electronic edition), 1998.

[4] Melis, E. et al: ActiveMath: A Generic and Adaptive Web-Based Learning Environment. International Journal of Artificial Intelligence in Education, Vol. 12, pp. 385–407, 2001.

[5] Alpert, S. R. et al: Deploying Intelligent Tutors on the Web: An Architecture and en Example. International Journal of Artificial Intelligence in Education, Vol. 10, pp. 183–197, 1999.

[6] Mitrović, A. and Hausler, K.: Porting SQL-Tutor to the Web. Proceedings of the International Workshop on Adaptive and Intelligent Web-based Educational Systems. Montreal, Canada, pp. 50–60, 2000.

[7] Devedžić, V. Understanding Ontological Engineering. Communications of the ACM, Vol 45, No. 4ve, pp. 136–144, 2002.

[8] Devedžić, V.: Next-Generation Web-Based Education. International Journal of Continuing Engineering Education and Life-Long Learning, Special Issue on The Issues of Technological Support for New Educational Perspectives (forthcoming), 2003.