

Doktorska disertacija
INTELIGENTNO PONAŠANJE SISTEMA ZA
UPRAVLJANJE UČENJEM

Goran Šimić

Univerzitet Singidunum, 2008.godina

Sadržaj

1. Uvod.....	5
2. Problem istraživanja.....	7
2.1. Psihološke teorije učenja.....	7
2.1.1 Konstruktivizam.....	7
2.1.2 Biheviorizam (Teorija o ponašanju)	8
2.1.3 Piaget.....	9
2.1.4 Vygotsky i socijalno saznanje.....	9
2.1.5 Stilovi učenja	10
2.1.5.1 Myers-Briggs-ov indikator tipova.....	10
2.1.5.2 Kolb-ijev eksperimentalni model učenja	12
2.1.5.3 Felder – Silverman model	13
2.1.6 Nastavni stilovi	16
2.1.6.1 Nastavni stilovi prema sadržaju	17
2.1.6.2 Nastavni stilovi prema prezentaciji.....	17
2.1.6.3 Nastavni stilovi prema učestvovanju studenata	17
2.1.6.4 Nastavnički stilovi prema perspektivi.....	18
2.1.7 Teorija višestruke inteligencije	18
2.1.8 Teorija kontrole.....	20
2.1.9 Neurologija	20
2.1.10 Teorija učenja fokusirana na ulogu mozga (Brain-based Learning).....	20
2.2. Modelovanje korisnika.....	21
2.2.1 Uopšte o korisničkim modelima	22
2.2.2 Tipologizacija korisničkih modela.....	24
2.2.2.1 Prekrivajući korisnički model	24
2.2.2.2 Stereotipni model	25
2.2.2.3 Implicitni i eksplicitni modeli	26
3. Analiza	27
3.1 Studentski modeli.....	27
3.1.1 IEEE PAPI (Public and Private Information) Learner.....	28
3.1.2 IMS LIP (Learner Information Package).....	30
3.1.3 EduPerson	36
3.1.4 Studentski model ELENA projekta	39
3.2 Modelovanje i pakovanje sadržaja učenja	41
3.2.1 Dublin Core.....	42
3.2.2 LOM (Learning Object Metamodel).....	46
3.2.3 IMS standardi za sadržaje učenja.....	48
3.2.3.1 IMS Learning Resources Meta-Data Specification	49
3.2.3.2 IMS Content Packaging	50
3.2.3.3 IMS Simple Sequencing	52
3.2.3.4 IMS Question & Test Interoperability	53
3.2.4 SCORM standard	56
3.2.4.1 SCORM model agregiranja sadržaja	58
3.2.4.2 Činioci (Assets)	58
3.2.4.3 Deljivi objekti sadržaja učenja (<i>Sharable Content Objects – SCOs</i>).....	58
3.2.4.4 Aktivnosti (<i>Activities</i>)	59

3.2.4.5 Organizacija sadržaja (<i>Content Organization</i>)	60
3.2.4.6 Agregacija sadržaja (<i>Content Aggregations</i>)	61
3.3 Sistemi za elektronsko učenje – kratak istorijat.....	63
3.3.1 Generativni sistemi	63
3.3.2 Adaptivno vežbanje i praksa.....	63
3.3.3 Kriza edukacione psihologije.....	63
3.3.4 Veštačka inteligencija i sistemi za elektronsko učenje	64
3.3.5 Uticaj kognitivnih nauka.....	65
3.3.6 Problem esencijalnog modela učenja.....	65
3.3.7 Socijalno-konstruktivistička tranzicija.....	66
3.3.8 Sadašnji trendovi.....	66
3.4 Adaptivni sistemi za elektronsko učenje.....	67
3.4.1 Inteligentni tutorski sistemi.....	68
3.4.1.1 Ekspertska modul	68
3.4.1.2 Domensko znanje.....	69
3.4.1.3 Pedagoški modul	70
3.4.1.4 Studentski model.....	72
3.4.1.5 Komunikacioni model.....	73
3.4.2 Primeri inteligentnih tutorskih sistema	74
3.4.2.1 Cognitive Tutor	74
3.4.2.2 SQL Tutor	76
3.4.3 Adaptivni edukativni hipermedijalni sistemi	78
3.4.3.1 Adaptacija predstavljanja sadržaja.....	79
3.4.3.2 Adaptacija navigacije.....	80
3.4.4 Primeri adaptivnih edukativnih hipermedijalnih sistema	81
3.4.4.1 ELMART	81
3.4.4.2 AHA!.....	84
3.4.4.3 InterBook	85
3.5 Sistemi za upravljanje učenjem	86
3.5.1 Moodle	88
3.5.2 WebCT i Blackboard	90
4 Prikaz predloženog rešenja	92
4.1 Arhitektura sistema	94
4.2 Izbor tehnologija	95
4.2.1 Prezentacioni sloj	97
4.2.2 Sloj servisa	98
4.2.3 Sloj poslovne logike.....	101
4.2.4 Sloj podataka.....	101
4.3 Prethodna istraživanja	102
4.3.1 Code Tutor	102
4.3.2 Multitutor	103
4.3.3 UNIS.Multitutor.....	104
5. Nastavnička aplikacija	105
5.3 Dizajn i implementacija	105
5.3.1 Slučajevi korišćenja	106
5.3.1.1 Upravljanje predmetima.....	107

5.3.1.2 Upravljanje sadržajima učenja.....	109
5.3.1.3 Upravljanje studentskim podacima.....	110
5.3.1.4 Saradnja sa drugim korisnicima.....	112
5.3.2 Konceptualni modeli nastavničke aplikacije	112
5.3.2.1 Paket <i>ContentManagmenetTool</i>	114
5.3.2.2 Paket <i>CourseManagmenetTool</i>	114
5.3.2.3 Paket <i>ColaborationTool</i>	115
5.3.2.4 Paket <i>DataBrokers</i>	116
5.3.2.5 Paket <i>AssessmentTool</i>	117
5.4 Napredno ocenjivanje	120
5.4.3 Korišćenje fuzzy logike u sistemu	122
5.4.3.1 Korišćeni koncepti	122
5.4.3.2 Tipovi fuzzy skupova.....	123
5.4.3.3 Fazifikacija.....	128
5.4.3.4 Fuzzy pravila i njihova i nicijalizacija	129
5.4.3.5 Fuzzy rezonovanje	130
5.4.3.6 Defazifikacija.....	133
5.4.5 Paketi fuzzy baze znanja.....	135
5.4.5.1 Paket <i>FuzzyOntology</i>	135
5.4.5.2 Paket <i>FuzzyReasoner</i>	136
5.4.5.3 Paket <i>FuzzyEvaluator</i>	137
5.4.5.4 Paket <i>DataBrokers</i>	138
5.4.6 Dijagrami interakcije	139
5.4.6.1 Fuzzy rezonovanje	139
5.4.6.2 Evaluacija fuzzy baze znanja	142
5.4.7 Korišćenje sistema: modelovanje Fuzzy baze znanja.....	143
5.4.7.1 Dijagram aktivnosti.....	143
5.4.7.2 Korisnički interfejs za definisanje termina	145
5.4.7.3 Implementacioni detalji vizuelizacije fuzzy baze znanja.....	146
5.4.7.4 Korisnički interfejs za definisanje pravila ocenjivanja	150
5.4.7.5 Implementacioni detalji fuzzy rezonovanja	151
5.5 Definisanje relacija između predmetnih činilaca.....	153
5.5.1 Vizualizator raspoređivanja (<i>Sequencing Visualizator</i>).....	154
5.5.2 Implementacioni detalji vizuelizacije povezivanja činilaca	156
6 Studentska aplikacija	158
6.1 Dizajn i implementacija	158
6.1.1 Slučajevi korišćenja	158
6.1.1.1 Učenje	159
6.1.1.2 Proveravanje	161
6.1.1.3 Upravljanje podsetnikom	163
6.1.2 Konceptualni modeli studentske aplikacije	163
6.1.2.1 Paket <i>ContentAndNavigationTool</i>	164
6.1.2.2 Paket <i>AssessmentTool</i>	166
6.1.2.3 Paket <i>CollaborationTool</i>	168
6.1.2.4 Paket <i>ReminderTool</i>	168
6.1.2.5 Paket <i>SudentModel</i>	169

6.1.2.6 Paket <i>DataBrokers</i>	172
6.1.3 Dijagram aktivnosti studentske aplikacije	172
6.1.3.1 Aktivnosti učenja	174
6.1.3.2 Kolaborativne aktivnosti	174
6.1.3.3 Aktivnost proveravanja	175
6.2 Adaptivna navigacija i sadržaji	175
6.2.1 Parametri aktivnosti	177
6.2.2 Dizajn u cilju adaptivnosti	180
6.2.3 Proširivanje standardne strukture sadržaja učenja	181
6.2.3.1 Dijagram aktivnosti	182
6.2.4 Šabloni i činjenice – senke	183
6.2.5 Baza pravila	185
6.2.5.1 Primer toka zaključivanja za rešavanje adaptivne navigacije	186
6.2.6 Dinamika zaključivanja	188
6.2.6.1 Instanciranje novih činjenica iz klijentske aplikacije	189
6.2.6.2 Kontrola radne memorije mašine za zaključivanje	190
7. Evaluacija	191
7.1 Adaptivno (fuzzy) ocenjivanje	191
7.1.1 Nedostaci adaptivnog ocenjivanja	193
7.1.2 Prednosti adaptivnog ocenjivanja	194
7.2 Adaptivna navigacija i sadržaji	196
7.2.1 Adaptivna navigacija	196
7.2.2 Adaptivni sadržaji	197
7.3 Upoređenje sa srodnim sistemima za e-učenje	198
7.3.1 Inovativno rešenje za sinhronu komunikaciju	200
8. Zaključak	201
9. Literatura	206
10. Skraćenice	214

1. Uvod

Učenje podržano računarima (*Computer Based Learning*), učenje na daljinu (*Distance Learning*), Web učionice (*Web Classrooms*), virtuelne učionice (*Virtual Classrooms*), predstavljaju deo terminologije kojima se opisuju sistemi za elektronsko učenje (*E-learning Systems*). Iako naziv *elektronsko* ima mnogo veću semantičku širinu, danas se u ovu kategoriju ubrajaju sistemi koji omogućavaju korisnicima da kreiraju, organizuju, kontrolišu i koriste sadržaje učenja i ostvaruju uzajamnu saradnju posredstvom računara i računarskih mreža.

Jedan od savremenih trendova u oblasti obrazovanja je individualistički pristup procesu učenja. To znači da edukativni sistem treba da ima znanje o svakom pojedinačnom studentu i mogućnost da prilagodi nastavne sadržaje njegovim konkretnim potrebama. Na žalost, najrasprostranjeniji sistemi za e-učenje – tzv. sistemi za upravljanje učenjem (*Learnign Management Systems – LMS*) nemaju, ili poseduju vrlo ograničenu mogućnost personalizacije procesa učenja.

Nasuprot ovim sistemima postoje adaptivni, ali domenski usko specijalizovani sistemi koji imaju svojstvo personalizacije učenja. Zbog toga se, adaptivni sistemi, u koje spadaju adaptivni hipermedijalni sistemi za edukaciju (*Adaptive Educational Hypermedia - AEH*) i inteligentni tutorski sistemi (*Intelligent Tutoring Systems - ITS*), još nazivaju inteligentnima. Manifestacije inteligencije u takvim sistemima su različite, ali je suština ista: inteligencija sistema je u tome da se sistem dinamički prilagođava potrebama pojedinca. Za razliku od tradicionalističkog pristupa učenju, u kome se studenti prilagođavaju kontekstu učenja i gde su manifestacije sistema podjednake za svakog studenta bez obzira na pojedinačne razlike i sklonosti, inteligentni sistemi ispoljavaju mnogo fleksibilnija svojstva. Načini sticanja znanja su ono što se menja i prilagođava korisniku sistema i u tom smislu inteligencija se može meriti stepenom adaptibilnosti u odnosu na individualnog korisnika.

Ovaj rad ima za cilj da izvrši analizu mogućnosti, prikaže teorijsko i praktično rešenje koje bi omogućilo fuzionisanje prednosti sistema za upravljanje učenjem (velika paleta funkcionalnosti za izvođenje nastave, standardizovanost, modularan dizajn i interoperabilnost) s jedne strane i adaptivnih svojstava inteligentnih sistema (dinamičko komponovanje i adaptacija navigacije i sadržaja učenja) s druge strane.

Pored ovog, rad sadrži još sedam poglavlja. Sadržaj se menja od čisto teorijskog na početku, do praktične implementacije i evaluacije sistema na kraju rada. Obimnost rada je bila uslovljena kompleksnošću tematike, njenom interdisciplinarnošću i velikom heterogenošću sistema koji se bave e-učenjem. Pored toga, obim rada je uslovljen i velikim brojem ilustracija, tabelarnih pregleda i kodnih fragmenata. Sledi kratak opis sadržaja.

U sledećem poglavlju je detaljan pregled teorijskih koncepata fenomenologije učenja. Predstavljene su psihološke teorije na kojima se temelji intelligentno ponašanje sistema u interakciji sa korisnikom. Prikazani su teorijski najznačajniji i najviše referencirani stilovi učenja i s njima dovedeni u vezu, nastavnički stilovi. Predstavljani su najvažniji tipovi korisničkih modela (aspekt personalizacije sadržaja): prekrivajući model i stereotipni

model. Objašnjeni su načini akvizicije podataka korisnika. Na osnovu iskustava iz prakse predložena je dodatna podela na implicitne i eksplicitne modele, sa obrazloženjem.

Treće poglavlje se bavi analizom praktičnih dostignuća iz oblasti e-učenja. Predstavljani su aktuelni standardi za struktuiranje studentskog modela i standardi za modelovanje i pakovanje sadržaja učenja. Za svaki od opisanih standarda dat je kritički osvrt na prednosti i slabosti. Takođe, predstavljani su odnosi između pojedinih standarda, kako bi se ukazalo na način njihovog evaluiranja kroz različite verzije. U drugom delu ovog poglavlja napravljen je kratak osvrt na razvoj sistema za e-učenje, neophodan za shvatanje savremenih trendova. Nakon toga predstavljani su reprezentativni primeri za svaku od navedenih vrsta: adaptivni hipermedijalni sistemi za edukaciju, inteligentni tutorski sistemi i sistemi za upravljanje učenjem. Opisana je namena, karakteristike sistema i različitost arhitekturnih rešenja.

U četvrtom poglavlju je prikazano predloženo rešenje: arhitektura sistema i izbor tehnologija. Pored toga prikazani su rezultati (iskustva) prethodnih istraživanja, a koja su u tesnoj vezi sa doktorskim radom. Akcenat je stavljen na izbor tehnologija za klijentski deo sistema, pošto su komponente serverskog dela detaljno razmatrane u narednim poglavljima.

Peto i šesto poglavlje bavi se problemima dizajna i implementacije predloženog rešenja. Dizajniranje je obavljeno korišćenjem UML jezika i metodologije za modelovanje procesa i entiteta. Funkcionalna dekompozicija je izvršena kroz dijagrame slučajeva korišćenja. Osnovne relacije i struktuiranje entiteta u sistemu definisane su preko konceptualnih dijagrama. Interakcije između objekata opisane su sekvencijalnim dijagramima, a tokovi aktivnosti – dijagramima aktivnosti.

Najpre je predstavljena nastavnička aplikacija (poglavlje 5). Težište ovog poglavlja je na primeni fuzzy logike u implementaciji naprednog (adaptivnog) ocenjivanja. Objašnjeni su fuzzy koncepti korišćeni za neegzaktno rezonovanje i predstavljena je ontologija preko koje su izražene njihove relacije. Predstavljen je metodološki pristup u realizaciji sistema sa fuzzy rezonovanjem. Objašnjene su različite tehnike pojedinih faza fuzzy rezonovanja. Pored prikaza dizajna i implementacije, predstavljen je način korišćenja sistema za adaptivno ocenjivanje.

U šestom poglavlju je predstavljena studentska aplikacija, na isti metodološki način kao u slučaju nastavničke aplikacije. Težište ovog poglavlja je na primeni diskretne logike i ekspertskih sistema radi postizanja navigacione i sadržajne adaptivnosti. Predstavljen je metodološki pristup u dizajnu baze znanja ekspertskog sistema i izvršena je formalizacija strukture baze pravila za potrebe navedenih adaptivnosti. Pored toga, prikazan je optimiziran način korišćenja radne memorije mašine za rezonovanje.

Sedmo poglavlje sadrži evaluaciju predloženog rešenja. Fokus evaluacije su adaptivna svojstva nastavničke i studentske aplikacije. Predstavljen je tabelarni pregled – komparacija karakteristika predloženog rešenja i aktuelnih sistema za upravljanje učenjem. Na kraju ovog materijala je zaključak u kome su istaknuti naučni doprinos, praktični doprinos rada i date smernice za dalji razvoj.

2. Problem istraživanja

Glavni problem kojim se bavi ovaj rad je: *kako personalizovati učenje u sistemima za upravljanje učenjem?* Personalizacija učenja znači da se sadržaji učenja, tempo i pedagoški aspekti, koje sistem ispoljava tokom korisničkih sesija, prilagođava svakom pojedinačnom studentu ponaosob. Ljudi uče na vrlo različite načine, o čemu govore psihološke teorije učenja.

2.1. Psihološke teorije učenja

Teorije i strategije učenja, kognitivni stilovi i stilovi učenja sa akcentom na teoriju višestruke inteligencije obezbeđuju teorijsku i praktičnu osnovu za personalizaciju procesa učenja. Postoji veliki broj teorija učenja. Zajedničko za svaku od njih je originalni pristup fenomenu učenja. Mogle bi se podeliti na ekstrospektivne i introspektivne [1][2]. Ekstrospektivne posmatraju učenje kao socijalni fenomen u kome individua, kroz razne vidove socijalnih interakcija spoznaje okruženje u kom se nalazi. U ekstrospektivne teorije mogli bi svrstati konstruktivističku i biheviorističku teoriju, Piaget-ovu teoriju i Vigotsky-jevu socijalno- saznavnu teoriju.

Introspektivne teorije opisuju učenje kao intrapersonalne procese, na koje dominantan uticaj ima psihofizička struktura ličnosti, lični motivi, sposobnosti i sklonosti. Introspektivni teoretičari svrstavaju socijalne uticaje na procese učenja u efemerne faktore. Ove teorije su više oslonjene na dostignuća egzaktnih nauka – psihologije, a naročito medicine. Najznačajnije su neurološka teorija, teorija o funkcionalnosti i strukturi mozga, teorija stilova učenja, teorija o višestrukoj inteligenciji i teorija kontrole. Naredni pasusi predstavljaju kratak pregled najznačajnijih teorija učenja.

2.1.1 Konstruktivizam

Konstruktivizam je teorija učenja zasnovana na stavu da čovek gradi shvatanje o svetu koji ga okružuje na osnovu iskustva [1][2]. Svaki čovek formira vlastita pravila i mentalne modele kojima doživljava ta iskustva. Učenje je opisano kao građenje mentalnih modela na osnovu novih iskustava. Postoji nekoliko najvažnijih principa konstruktivizma. Učenje je misaono pretraživanje. Zato, učenje mora početi sa problemima koje studente zaista zaokupljaju. Sledeći princip je da učenje zahteva razumevanje celine kao i njenih delova. Delovi moraju biti shvaćeni u kontekstu celine. Tako da učenje mora bit fokusirano na osnovne koncepte a ne izolovane činjenice. Da bi dobro držali nastavu, moramo shvatati mentalne modele kako studenti shvataju svet i shvatanja koja usvajaju radi podrške njihovom modelu. Pri učenju pojedinac konstruiše vlastito mišljenje (ne samo memorisanje pravih odgovora i prihvatanje tuđeg mišljenja). Pošto je edukacija interdisciplinarno nasleđe, jedini merljivi način za ocenu učenja je da vrednovanje postane deo procesa učenja (učenje rešavanjem problema - *problem based learning*), deo koji pre svega treba da pokaže studentima koliko znaju i šta treba bolje da znaju.

Konstruktivisti smatraju da nastavne programe treba učiniti adaptibilnim i prilagođenim individualnim saznavnim potrebama studenata. Nastavnik bira strategiju na osnovu reakcija i rezultata studenta, ohrabrujući ga da analizira, interpretira i predviđa informacije (činjenice). Radi što bolje efektivnosti problemskog učenja, konstruktivizam

promoviše ne samo komunikaciju nastavnik-student, već i studenata međusobno (kolaborativno učenje).

Pored učenja na osnovu problema, situaciono učenje predstavlja još jednu aplikaciju konstruktivističke epistemologije. Polazi se od činjenice da znanje ne predstavljaju statičke simboličke reprezentacije smeštene u mozgu, već je situirano, i predstavlja produkt aktivnosti, konteksta i kulture u kojoj se razvijalo i u kojoj se koristi. Klasičan primer situacionog učenja je učenje jezika: deca uče jezik neverovatnom brzinom zahvaljujući prisutnosti u sredini u kojoj se isti jezik koristi. S druge strane, strani jezik se u školama uči godinama, i većina participirajućih ga nikad ne nauči, jer ga u praksi ne koristi.

Konstruktivizam poziva na izbacivanje rangiranja i standardizovanog testiranja. Umesto toga, ocenjivanje treba da postane deo nastave, tako da studenti imaju veću odgovornost u ocenjivanju njihovog vlastitog progressa.

2.1.2 Biheviorizam (Teorija o ponašanju)

Biheviorizam je teorija učenja koja je fokusirana samo na objektivno očigledno ponašanje [1]. Mentalne aktivnosti su u potpunosti isključene iz razmatranja. Za bihevioriste učenje predstavlja akviziciju novog ponašanja [1][2]. Eksperimenti biheviorista vide uslovljavanje kao univerzalni proces učenja. Diferencirali su dva tipa uslovljavanja, svaki sa posebnom mustrom ponašanja:

Klasično uslovljavanje – događa se u slučaju prirodne reakcije na stimulus. Najpopularniji primer je Pavlovljevi eksperimenti sa hranjenjem pasa u uslovima obojenog svetla. Psi se curela voda na usta u zavisnosti od osvetljenja i kada ne vide hranu. Pošto su životinje i ljudi biološki povezani, znači i kod ljudi će odgovarajući stimulus proizvesti predvidljiv odgovor.

Biheviorističko ili operantno uslovljavanje – desiće se kada je odgovor na stimulus pojačan. Prosto rečeno, operant je jednostavan sistem povratne sprege. Ako nagrada ili pojačavanje sledi odgovor na stimulus, odgovor će postati predvidljiv u budućnosti. Npr. vodeći biheviorista B.F. Skinner koristio je tehnike pojačavanja da nauči golubove da igraju i guraju loptice.

Postoji mnogo kritika biheviorizma na račun ograničenosti teorije samo na očigledno ponašanje. Biheviorizam ne može da objasni neke vrste učenja. Na primer, prepoznavanje šablona novog jezika kod male dece, gde nema mehanizma pojačavanja. Po mišljenjima mnogih, glavni nedostatak teorije je preterana smernost na uslovljavanje stimulusima.

Biheviorizam je teorija relativno jednostavna za razumevanje, jer počiva samo na opažljivom ponašanju i opisuje nekoliko univerzalnih zakona ponašanja. Tehnike pozitivnog i negativnog pojačavanja vrlo su efikasne za životinje, ali i za poremećene ljude kao što su autisti (i drugi primeri asocijalnih ponašanja). Očigledna aplikacija biheviorizma u nastavničkoj praksi je kažnjavanje studenata zbog njihovog nepoželjnog ponašanja.

2.1.3 Piaget

Švajcarski biolog i psiholog Jean Piaget (1896-1980) je poznat po konstrukciji modela visokog uticaja razvoja i učenja kod dece [1]. Ova teorija bazirana je na ideji da, razvijajući se, dete gradi kognitivne strukture: mentalne mape, šeme ili umrežene koncepte za razumevanje i odgovor na fizička iskustva iz sveta koji ga okružuje. Piaget je dalje potvrdio da se kognitivna struktura deteta uvećava i biva sofisticiranija počevši od nekoliko prirodnih refleksa do vrlo kompleksnih mentalnih aktivnosti. Piaget-ova teorija identifikuje 4 stadijuma razvoja dece:

1. Senzomotorni stadijum (rođenje - 2 godine starosti), u kome dete, kroz fizičku interakciju sa okruženjem izrađuje skup koncepata o realnosti i kako ona funkcioniše. To je stupanj u kome deca ne znaju da objekti i dalje postoje iako su im van vidokruga.
2. Preoperacionalni stadijum (starost 2-7), u kome dete još nije sposobno da konceptualizuje apstraktno.
3. Stadijum konkretnih operacija (starost 7-11) u kome dete počinje da, akumulirajući fizička iskustva, konceptualizuje kreirajući logičke strukture koje objašnjavaju njegovo iskustvo. Na ovom stupnju moguće je rešavanje apstraktnih problema. Aritmetičke jednačine sa brojevima mogu biti rešene ali ne i jednačine sa objektima.
4. Stadijum formalnih operacija (početak sazrevanja 11-15) u kome detetove kognitivne strukture liče na strukture odraslog i uključuju konceptualno zaključivanje.

Piaget je izdvojio nekoliko principa za razvoj kognitivnih struktura. Tokom svih faza razvoja, dete iskušava okruženje koristeći već konstruisane mentalne mape. Ako je iskustvo ponovljeno ono se smešta u kognitivnu strukturu. Ako je iskustvo novo onda se kognitivna struktura (equilibrium) menja prilagođavajući je novoj situaciji (iskustvu). Na ovaj način dete razvija sve više odgovarajućih kognitivnih struktura.

Piaget smatra da nastavni programi treba da omoguće i logički i konceptualni razvoj. Nastavnici moraju istaći kritičnu ulogu koju iskustvo ili interakcije sa okruženjem igraju u učenju studenta. Piaget-ova teorija dala je znatan naučni doprinos u razumevanju kognitivnih procesa, i ovi rezultati su ugrađeni i u teoriju konstruktivizma.

2.1.4 Vygotsky i socijalno saznanje

Ova teorija zastupa stav da je kultura osnovna determinanta individualnog razvoja. Ljudi su vrsta koja ima kulturu i intelektualni razvoj koji je u kontekstu te kulture (kulturno i porodično okruženje). Vygotski je proučavao fenomen učenja [1] kroz intelektualni razvoj najmlađih u okvirima različitih kulturoloških ambijenata.

Kultura ima dvojak uticaj na dečji intelektualni razvoj: kroz kulturu deca savlađuju skoro sve njihovo znanje. Kulturno okruženje daje deci okvire u kojima se odvijaju saznajni procesi, kako Vygotsky kaže - alatke intelektualne adaptacije. Kultura daje okvire šta i kako da se uči. Kognitivni razvoj rezultira iz dijalektičkog procesa kroz koji deca uče kroz iskustvo u rešavanju problema zajedno sa drugima (roditelj, nastavnik, drugovi i

rođaci). Inicijalno interakcija ljudi i dece započinje sa velikom odgovornošću starijih u rešavanju problema, ali postepeno se ta odgovornost prenosi na decu.

Mogućnosti samostalnog učenja i učenja uz pomoć su različite. Vygotsky tu razliku naziva zonom proksimalnog razvoja (razvoja pomoću socijalnih agenata). Pošto je veći deo onoga što dete uči iz kulturnog okruženja i veći deo rešavanja problema je uz pomoć punoletnih, nemoguće je posmatrati dete bez uticaja okoline. Pomoć starijih Vygotsky naziva skelama. One ne proizvode samo neposredne rezultate, već povećavaju veštine samostalnog rešavanja problema u budućnosti.

Prema teoriji socijalnog saznanja, ocenjivanje mora biti u zoni proksimalnog razvoja. Šta deca mogu samostalno da urade je njihov aktuelni nivo razvoja, a šta mogu uz pomoć drugih je potencijalni nivo razvoja. Dva deteta mogu imati isti nivo aktuelnog razvoja ali uz adekvatnu pomoć starijeg jedno će moći rešiti mnogo više problema od drugog deteta. Ocenjivanje mora da pokazuje i nivo aktuelnog i nivo potencijalnog razvoja.

2.1.5 Stilovi učenja

Studenti primaju i obrađuju informacije na vrlo različite načine. Teoretičari ovog pravca nastoje da izmere uticaj ličnih stilova učenja na efektivnost učenja. Stilovi učenja su “karakteristična kognitivna, afektivna, i psihološka ponašanja koja služe kao relativno stabilni indikatori za merenje kako studenti razumeju, interaguju sa i odgovaraju na okruženje za učenje” [4]. Koncept stilova učenja ukorenjen je u klasifikaciji psiholoških tipova [1]. Postoje mnogobrojne teorije stilova učenja. U ovom poglavlju su predstavljene tri teorije, koje su prošle praktičnu evaluaciju i koje se koriste u aktuelnim psihološkim, pedagoškim i andragoškim istraživanjima.

2.1.5.1 Myers-Briggs-ov indikator tipova

Jedna od najpoznatijih teorija učenja je Jung-ova teorija psiholoških tipova. Ova teorija je operacionalizovana kao model kroz Myers-Briggs-ov Indikator tipova (MBTI) [5]. Jung predstavlja četiri klasifikacije psiholoških tipova, koje imaju izrazite implikacije na učenje (Tabela 1).

Tabela 1: Myers-Briggs-ov Indikator tipova

KRITERIJUM	TIPOVI
Spoznaja	Ekstravertni
	Intravertni
Intuicija	Senzorni
	Intuitorni
Razmišljanje	Mislioci
	Osećajnici

Osećaji	Sudije
	Shvatači

Ekstravertni tip spoznaje (uči) fokusiran na spoljašnji svet i ljude koji ga okružuju. Sklon je izvođenju zaključaka na osnovu fenomena, a socijalan profil mu garantuje uspešnu interpersonalnu komunikaciju. *Intravertni* tipovi su okrenuti sebi, pokušavajući da spoznaju stvari na osnovu ličnih iskustava i ideja. Intravertni su time skloniji tradicionalnoj nastavi, individualnim obavezama, nego aktivnom učešću na časovima i u kooperativnom učenju

Senzorski tipovi su praktičari, usmereni na detalje, činjenice i procedure. *Intuitivni* tipovi su konceptualno orijentisani, fokusirani na značenja i mogućnosti i sa većom mogućnošću imaginacije. Intuitivni tipovi su zato više orijentisani na naučne fundamente.

Mislioni su skeptici koji donose odluke logično i na osnovu jasno definisanih pravila. *Osećajnici* nastoje da donose odluke na osnovu ličnih i humanističkih razmatranja. Oni su altruisti sa niskim pragom satisfakcije. Mislioni dakle, daju veći značaj objektivnoj analizi nego interpersonalnim razmatranjima u donošenjima odluka.

Sudije postavljaju i slede agende, tražeći detalje čak i u nepotpunim podacima. *Shvatači* trude se da promene okolnosti i odlože traganje za detaljima, radi obezbeđivanja više podataka. Sudije vide težište da rade prema silabusu i prema vremenskim rokovima obaveza, a istraživanje novih ideja i kreativnost u rešavanju problema su im strani.

U empirijskim longitudinalnim (duževremenskim) istraživanjima na osnovu MBTI, na grupi od 116 studenata koji su pohađali kurseve iz hemijskog inženjerstva (North Carolina State University) [6] došlo se do sledećih zaključaka:

- Intuitori su bolji od senzora u predmetima sa sadržajem visokog nivoa apstrakcije, a obrnuto je bilo zabeleženo u mnogo praktičnijim kursevima.
- Extravertni su reagovali mnogo pozitivnije nego introvertni kod zahteva da se domaći rad odvija u grupama. Na kraju kursa obe grupe su se ipak u potpunosti opredelile za grupni rad.
- Intuitori su tri puta više od senzora davali najviše ocene kod vrednovanja sposobnosti za rešavanje problema i davali veliki značaj kreativnom radu u tokom karijere.
- Većina senzora je nastojalo da se zaposli u velikim korporacijama, dok je mnogo veći procenat intuitora planirao da radi u manjim kompanijama, da nastavi školovanje, ili da radi na istraživanjima. Osećajnici su više vrednovali socijalno značajan i dobrotvoran rad u njihovim karijerama nego što su to činili mislioci.

Konačna ocena MBTI je da je vrlo upotrebljiv za otkrivanje razlika u načinima na koje studenti pristupaju zadacima učenja, kako reaguju na različite forme podučavanja i različita nastavna okruženja i kako formulišu svoje ciljeve u karijeri.

2.1.5.2 Kolb-ijev eksperimentalni model učenja

Jedna od češće pominjanih tipologizacija je Kolb-ijev model koji klasifikuje studente prema njihovim preferencama za prihvatanje i procesiranje informacija [8, 9]. Prijem informacija može biti:

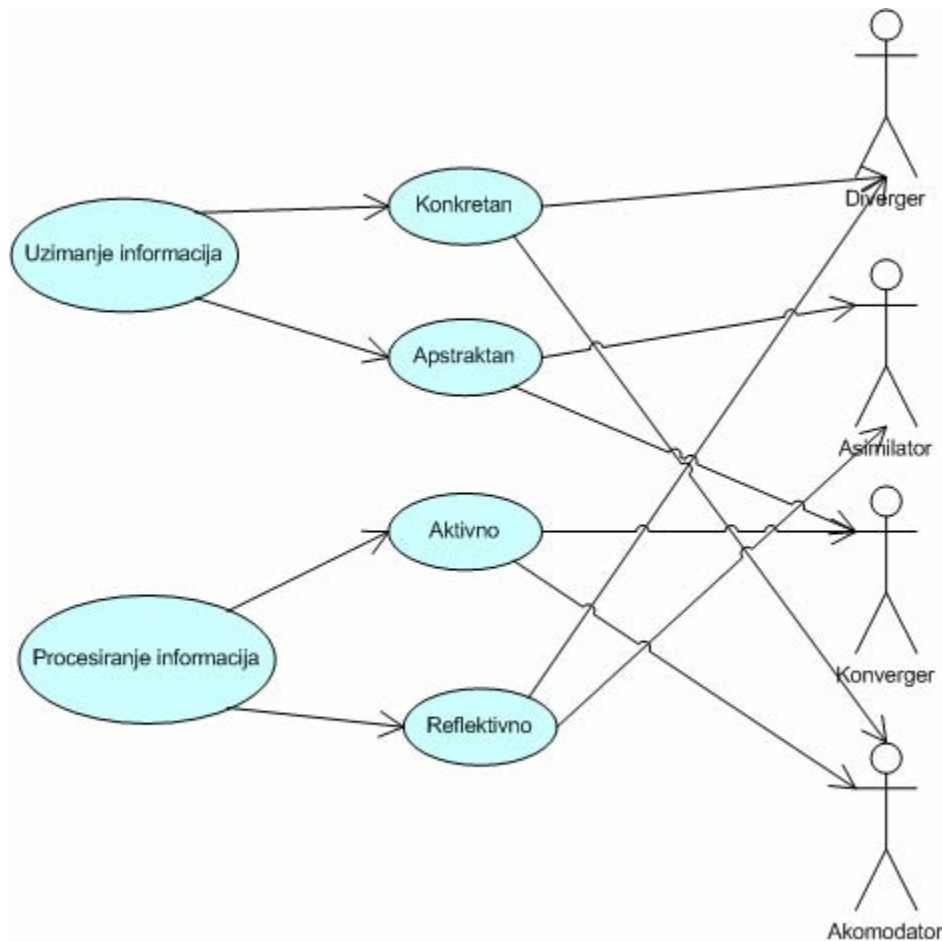
- Konkretni – konkretni stil bi značio prihvatanje informacija kroz direktno iskustvo, kroz opažanje, aktivnosti i osećaje
- Apstraktni – apstraktni stil bi značio prijem informacija kroz analize, praćenja i razmišljanja.

Prema procesiranju informacija postoje:

- Aktivni – aktivni procesori već u toku korišćenja novih informacija stiču iskustva;
- Reflektivni procesori – uče nakon završenog prijema i korišćenja informacija, razmišljajući o tome.

Prema ova dva kriterijuma diferenciraju se četiri stila učenja – tipa studenata (Ilustracija 1):

- *Diverger* – tip studenata koji često postavljaju pitanje *zašto?* Oni poseduju stil učenja kom odgovaraju objašnjenja u kakvoj je vezi nastavni materijal sa njihovim interesovanjima, iskustvima i budućom karijerom. Nastavnik dakle mora da bude motivacioni faktor.
- *Asimilator* – njihovo karakteristično pitanje je *šta?* Ovi studenti poseduju stil učenja kom odgovara logički jasno i dobro organizovano predstavljanje informacija. Asimilatori su u velikoj prednosti nad ostalim stilovima ako imaju vremena za refleksije. Nastavnik u ovom slučaju mora da bude pre svega ekspert.
- *Konvergor* – njihovo karakteristično pitanje je *kako?* Konvergori vole da uče metodama pokušaja i pogrešaka, što znači da okruženje treba da omogući *bezbedan pad*. Preferiraju aktivan rad na dobro definisanim zadacima. Da bi bio efektan, nastavnik preuzima ulogu trenera, koji vodi studente kroz praksis, pružajući povratne informacije za ispravljanje grešaka.
- *Akomodator* – njihovo karakteristično pitanje je *šta ako?* Fokusirani su na primenjivost nastavnih materijala u novim situacijama i na rešavanje problema. Za ovaj stil, nastavnik treba da ostavlja studentima otvorena pitanja, da ih sami rešavaju i istražuju stvari samostalno. Učenje rešavanjem problema (*Problem Based Learning*) predstavlja optimalnu pedagošku strategiju za ove studente.



Ilustracija 1: Kolb-ijev model stilova učenja

Nasleđe, vaspitanje i okolina utiču na različitost prihvatanja i obrade informacija kod pojedinaca. Teoretičari stilova učenja smatraju da nastavne metode treba da obuhvate sva četiri stila učenja. U nastavnom procesu potrebno je koristiti različite iskustvene (multimedijalne) sadržaje: zvuk, muziku, slike, film, iskustvo i pričanje. Da bi se mogao izraziti svaki od stilova učenja, ovi teoretičari zagovaraju korišćenje što više različitih tehnika ocenjivanja. Teorija stilova učenja ističe i značaj adaptibilnosti nastave (nastavnih sadržaja, nastavnih metoda i oblika rada, ocenjivanja). Kroz radove kritikuju tradicionalne škole, koje nastoje da favorizuju apstraktnu percepciju i reflektivno procesiranje informacija.

2.1.5.3 Felder – Silverman model

Felder – Silverman model je jedan od najviše eksploatisanih modela u aktualnim pedagoškim istraživanjima. Po ovom modelu studentov stil učenja može se definisati na osnovu odgovora na četiri pitanja:

- Koji tip informacija student favorizuje (prihvata)? Konkretne, ili apstraktne? Ova skala identična je senzo-intuitivnoj skali Myers-Briggs-ovg indikatora tipova.
- Koji tip senzora najefikasnije prihvata informacije? Vizuelni, ili verbalni?

- Kako student procesira informaciju? Aktivno, ili reflektivno? Ova skala identična je aktivno-reflektivnoj skali Kolb-ovog modela. Pored toga povezana je sa intravertno-ekstravertnom skalom Myers-Briggs-ovg indikatora tipova.
- Koja je studentova karakteristika napredovanja ka razumevanju materije? Sekvencijalna, ili globalna?

Navedena četiri pitanja zapravo imaju osnovu u četiri dimenzije, na osnovu kojih je Felder – Silverman model skaliran. Ova tipologizacija stilova učenja, je kreirana na osnovu psihološke ekspertize i iskustva u obrazovanju. Predstavljena je sledećom podelom (Tabela 2).

Tabela 2: Dimenzije stilova učenja (Felder – Silverman, 2005)

DIMENZIJE	STILOVI
Percepcija	Senzorni
	Intuitivni
Način prijema informacija	Vizuelni
	Verbalni
Procesiranje	Aktivno
	Reflektivno
Razumevanje	Sekvencijalno
	Globlano

Predstavljene su četiri dimenzije stilova učenja [3][7]: percepcija, način prihvatanja, organizacija, obrada i razumevanje.

Senzorni naspram intuitivnog stila

Dimenzija *percepcija* izvedena je na osnovu Jung-ovih psiholoških tipova. Senzorni tip radije spoznaje svet (uči) na osnovu direktnih opažaja iz okruženja. Intuitivni tip preferira indirektnu spoznaju na osnovu razmišljanja i imaginacije. Konkretnim ispitivanjima došlo se do zaključaka da senzorni tipovi vole činjenice, podatke i eksperimente. Intuitivni tipovi vole principe i teorije. Senzorni tipovi su strpljivi sa detaljima ali ne vole komplikacije. Oni vole rešavanje problema ali *bez iznenađenja*. Intuitivni se uspešno nose sa problemima ali su im detalji jako zamorni. Senzorni tipovi lako pamte činjenice, pažljivi su ali zato vrlo spori. Intuitivni tipovi brzo uče, prosto grabe nove koncepte, ali propuštajući mnoge detalje. Intuitivni tipovi se lako snalaze sa simbolima. Pošto su reči simboli, intuitivni tip ih lako prevodi u ono što zapravo predstavljaju. Spora translacija reči kod senzornog tipa čini ih vrlo neefikasnim kod vremenski ograničenih proveravanja.

Vizuelni naspram verbalnog stila

Vizuelne informacije uključuju slike, dijagrame, crteže, animacije i slično. Verbalne informacije su one koje su izrečene (govor) ili napisane. Vizuelni tip lako pamti vizualizovane informacije, dok vrlo lako zaboravlja one koje su mu samo verbalno saopštene, ili pročitane. Verbalni tipovi ispoljavaju potpuno drukčije sklonosti. Pored bržeg prihvatanja onog što im je verbalno saopšteno, pogoduje im učenje kroz razgovor, diskusije i objašnjavanje.

Aktivni naspram reflektivnog stila

Kompleksni mentalni procesi koji primljene informacije transformišu u znanje mogu se grupisati u dve kategorije – aktivno eksperimentisanje i reflektivna opservacija. Analogno perceptivnim procesima, postoje dva stila učenja u odnosu na ovo svojstvo: aktivni i reflektivni tip. Aktivno eksperimentisanje uključuje aktivnosti u realnom svetu na osnovu primljene informacije: diskusija, objašnjavanje, testiranje i slično. Aktivni tip zato ima više sklonosti ka radu u grupama (timski rad), a slaba im je efikasnost u situacijama kada imaju pasivnu ulogu u procesu učenja (što je najčešći slučaj kod tradicionalne nastave). Reflektivni tip je zapravo potpuni oponent aktivnom tipu. Ovaj tip je dobar manipulator informacijama – introspektivno, skloniji samostalnom radu, nego radu u grupama, i generalno tip iz kojeg će se najpre razviti dobar teoretičar nego praktičar. Aktivni i reflektivni stilovi su tesno povezani sa ekstravertim i introvertnim tipovima Jung-Myers-Briggs-ovog modela [5].

Sekvencijalni naspram globalnog stila

Formalna edukacija podrazumeva da nastavni materijali budu predstavljeni u logičkom, vremenski definisanom sledu. Pošto studenti prođu nastavne materijale, bivaju obično testirani, nakon čega prelaze na novu tematsku celinu (materijale). Za studente, kojima odgovara ovakav način učenja, kaže se da uče *sekvencijalno* – imaju sekvencijalni stil. Drugi – globalni stil poseduju studenti koji imaju izraženi skepticizam i treba im duže vremena, ili više gradiva da usvoje razumevanje rudimentarnih koncepata. Oni gube dane i nedelje učeći, shvatajući osnovne probleme. Ali, kada napokon shvate, ili pronađu rešenje (iluminacija), izuzetno brzo usvajaju svo preostalo gradivo. Oni tada postaju sposobni da primene usvojeno znanje na praktične probleme, mnogo efikasnije nego studenti koji uče sekvencijalno.

Sekvencijalni studenti pri rešavanju problema slede linearni proces rezonovanja. Oni se dobro snalaze sa materijalom koji poznaju delimično ili površno. Njihova dobra osobina je da su više skloni konvergentnom razmišljanju i analizi nego globalni. Sekvencijalni studenti najbolje uče kada je materijal prezentovan kao stabilna progresija kompleksnosti i težine.

Globalni stil učenja omogućava intuitivne skokove pri rešavanju problema, često bez mogućnosti eksplicitnog objašnjavanja kako se došlo do rešenja. Globalni stil učenja preferira divergentno razmišljanje i sintezu. Tradicionalno školovanje ne pogoduje globalnom stilu, jer studenti ovog stila postižu bolje rezultate ako im se omogući direktan pristup kompleksnijim i težim materijalima. Globalnim studentima visoko obrazovanje ne bi trebalo da bude problem, oni su sintenzatori, multidisciplinarni istraživači koji mogu da uoče zavisnosti koje drugi ne vide, a ako prežive edukacioni proces izrastaju u vrhunske eksperte.

2.1.6 Nastavni stilovi

Navedene kategorizacije stilova učenja ukazuju na kompleksnost procesa učenja. One definišu čiste stereotipe kao apstraktne definicije. Čisti stereotipi u stvarnosti ne postoje. Kod realnih studenata oni se uzajamno prepliću. Stereotipi su fabrikovani radi diferenciranja stilova učenja. Postojanje više dimenzija (kriterijuma) razmatranja (različiti stilovi u odnosu na različite kriterijume razmatranja) još više usložnjava problematiku učenja. Nastavnički stilovi predstavljaju paralelu stilovima učenja. Nastavnici treba da odgovore različitim stilovima primenom različitih pedagoških strategija i konkretnih nastavnih taktika. Uzimajući u razmatranje Felder-Silverman model stilova učenja, koji sublimira istraživanja u svim prethodnim modelima, nastavnički stilovi se mogu takođe klasifikovati na osnovu četiri kriterijuma (dimenzije): vrste sadržaja, načina prikazivanja, načina na koji studenti učestvuju u nastavi, i na osnovu perspektive iz koje studenti vide materiju za učenje (Tabela 3).

Tabela 3: Nastavni stilovi (Felder-Silverman)

Preferirani stil učenja		Odgovarajući nastavni stil	
Percepcija	Senzorni	Konkretni	Sadržaj
	Intuitivni	Apstraktni	
Način prijema informacija	Vizuelni	Vizuelni	Prezentacija
	Verbalni	Verbalni	
Procesiranje	Aktivno	Aktivno	Učestvovanje studenata
	Reflektivno	Pasivno	
Razumevanje	Sekvencijalno	Sekvencijalno	Perspektiva
	Globlano	Globlano	

Nastavnički stil se može definisati na osnovu odgovora na 5 pitanja:

- Koji tip informacije je istaknut od strane nastavnika: konkretan – faktički, ili apstraktan – konceptualan, teoretski?
- Koji način predstavljanja je istaknut: vizuelni – slike, dijagrami, video, demonstracije, ili verbalni – lekcije, čitanje, diskusije?
- Koji je način učestvovanja studenata ostvaren u toku izvođenja nastave: aktivno – studenti razgovaraju, kreću se, reflektuju znanje, ili pasivno – studenti samo gledaju i slušaju?
- Kakva je perspektiva na prezentovane informacije: sekvencijalna – napredovanje korak po korak, ili globalna – kontekst i relevantnost?

Pored navedenih dimenzija nastavničkih (i studentskih) stilova postoji još jedna, koja je uklonjena iz razmatranja [12]. Organizaciona dimenzija sadrži dva karakteristična stila: *induktivni* i *deduktivni*. Kod induktivnog stila fenomeni vode prema principima (od konkretnog prema apstraktnom), dok se kod deduktivnog stila dešava obratno. Ovi stilovi imaju suštinski različita značenja u zavisnosti na perspektivu (nastavničku ili studentsku). Autori preferiraju da je indukcija najbolji nastavnički metod, bilo da se zove učenje rešavanjem problema, učenje istraživanjem, ili ispitivanjem. S druge strane dedukcija je najčešće primenjivan metod učenja u tradicionalnom školskom sistemu.

Problem sa induktivnom organizacijom nastave je što nije koncizna i nema jasnu perspektivu (zapravo nameru). Bez obzira koliko se nastavniku njegova prezentacija činila nedvosmislena ili intuitivna, student nema jasnu sliku i motiv učenja; koje prednosti ima rešavajući problem, sakupljajući i obrađujući veliku količinu opservacija, ili podataka? Većina studenata na konsultacijama izjavljuje: *recite tačno šta treba da znam za test, ni reč više ili manje*. Ovaj studentski pristup nastavi je tipično *deduktivan*, i ohrabruje nastavnike da zadrže tradicionalne paradigme držanja nastave, iako su se u praksi pokazale kao manje efektne od induktivnih. Iz navedenih razloga organizaciona dimenzija definisanja nastavničkih stilova uklonjena je iz razmatranja.

2.1.6.1 Nastavni stilovi prema sadržaju

Podela stilova na *konkretan* i *apstraktan* veoma podseća na Kolb-ijev model, koji prema percepciji informacija u procesu učenja deli stilove na apstraktne i konkretne. Ova dva nastavna stila odgovaraju senzorskim i intuitivnim stilovima učenja. Pošto senzorski tip studenata lakše saznaje svet na osnovu konkretnih opažaja, činjenica, podataka i eksperimenata, njima više odgovara konkretan nastavni stil, koji tačno nudi navedene tipove nastavnih sadržaja. Apstraktan nastavni stil u prvi plan stavlja sublimirano teorijsko znanje, principe i koncepte, koji su oslobođeni velikog broja praktičnih detalja i konkretnih činjenica, koje inače opterećuju učenje studenta intuitivnog stila. Drugim rečima, sadržajno, apstraktni nastavni stil odgovara intuitivnom stilu učenja.

2.1.6.2 Nastavni stilovi prema prezentaciji

Prema dimenziji prezentacije, nastavnički stilovi se u potpunosti podudaraju sa stilovima učenja prema preferiranom načinu prijema informacija: *verbalni* i *vizuelni* stil. Vizuelni stil nastave uključuje multimedijalno prezentovanje sadržaja, koje zahteva korišćenje savremenih tehnologija. Prenosni medijumi su: slike, dijagrami, crteži, video i audio materijali. Verbalni nastavni stil se oslanja na saopštavanje informacija direktnim govorom ili kroz tekstualne materijale. Verbalni stil učenja podrazumeva obostranu razmenu informacija između nastavnika i studenata: diskusije, razgovori, konsultacije i objašnjavanja. Ovaj stil je stariji i ne zahteva učešće tehnologije, izuzev za različite interaktivne forme u sistemima za učenje na daljinu: forumi, diskusione grupe, tematske *chat* sesije i slično.

2.1.6.3 Nastavni stilovi prema učestvovanju studenata

Aktivni i *pasivni* nastavni stilovi predstavljaju grubu podelu na osnovu prirode učestvovanja studenata u nastavi. *Aktivni* stil znači da studenti obavljaju još neke aktivnosti u toku nastave, izuzev jednostavnog slušanja i gledanja. One mogu biti organizovane u vidu diskusija, postavljanja pitanja (ispitivanja), argumentovanja, razmene informacija u vezi nastavne građe (*brainstorming*), rada na rešavanju zadataka i problema, ili refleksije vlastitog znanja i razumevanja drugima (seminarski radovi, referati).

Ova dimenzija nastavnih stilova odgovara dimenziji procesiranja informacija kod stilova učenja (aktivan i reflektivni stil učenja). Međutim, aktivno učestvovanje studenata u nastavi je usmereno i na procese učenja kroz aktivnu eksperimentaciju i kroz reflektivnu opservaciju. To znači da, na nastavi u kojoj su svi studenti uvek *pasivni*, niti učenje

aktivnih eksperimentatora, niti reflektivnih opservatora nije efektivno. Na žalost veliki deo nastave tradicionalnog školovanja spada u tu kategoriju. Nastavnici treba da po svaku cenu izbegnu da se njihov stil okarakteriša kao pasivni nastavni stil.

2.1.6.4 Nastavnički stilovi prema perspektivi

Prema kriterijumu perspektive, postoje *globalni* i *sekvencijalni* nastavni stil. Ovi nastavni stilovi se uklapaju u istoimene stilove učenja po kriterijumu razumevanja. Sekvencijalni način nastave (postupna) predstavlja tradicionalni i opšte prihvaćen način držanja nastave. Kurikulumi, silabusi predmeta, udžbenici su sekvencijalni, kao i organizacija čitavog školskog sistema. Na taj način sekvencijalni stil je postao dominantan nastavni stil.

Međutim, postavlja se pitanje kako učiti studente globalnog stila učenja? Da bi se to postiglo, nastavnici treba da prikažu studentima *veliku sliku*, ili cilj lekcije pre nego što pređu na konkretne nastavne korake. Na taj način postavljaju konkretnu nastavnu celinu u globalni kontekst i omogućavaju studentima da je povežu sa prethodnim saznanjima, što uzrokuje bolje razumevanje i motivaciju.

Uključivanjem što više kreativnih vežbanja u nastavu, kao što je rešavanje problema, nastavnik će vlastitim globalnim stilom postići značajni napredak i kod globalnih i kod sekvencijalnih studenata. Studenti će biti u situaciji da koriste znanja iz različitih predmeta, iz prethodne edukacije, kako bi našli rešenja za zadate probleme. Još jedna taktika globalnog stila može proizvesti efektne rezultate – globalnim studentima objasniti njihov mehanizam učenja. Na taj način biće manje samokritični (jer se ne uklapaju u opšte prihvaćenu sekvencijalnost), lakše će se snalaziti u nastavnoj materiji, i ostvariće mnogo pozitivniji stav o edukaciji uopšte.

2.1.7 Teorija višestruke inteligencije

Psiholog Howard Gardner tvrdi da postoje najmanje 7 načina na koji ljudsko biće spoznaje svet. On ih označava kao različite inteligencije [1]. Inteligencije su veštine koje obezbeđuju individui da rešava stvarne probleme sa kojima se sučeljava. Gardner definiše inteligenciju kao grupu svojstava koja se razlikuje se od ostalih ljudskih kapaciteta, ima suštinski set operacija za obradu informacija, koja se razvija na uniformni način kod svih ljudi i koja je čvrsto ukorenjena u evoluciji čoveka kako razumskog bića.

Gardner sugeriša da lista inteligencija nije konačna, mada daje 7 prepoznatljivih tipova:

- *Verbalno-lingvistička* - sposobnost korišćenja reči i jezika.
- *Logičko-matematička* - kapacitet induktivnog i deduktivnog razmišljanja i rezonovanja, sposobnost korišćenja brojeva i prepoznavanja apstraktnih šablona.
- *Vizuelno-prostorna* - sposobnost vizualizacije objekata i prostornih dimenzija i kreiranja imaginarnih slika i predstava.
- *Telesno-kinetička* - mudrost tela i sposobnost kontrole pokreta.
- *Muzičko-ritmična* - sposobnost prepoznavanja zvučnih sekvenci i zvukova i osećaj za ritam.

- *Interpersonalna* - sposobnost komunikacije i relacije između osoba.
- *Intrapersonalna* - unutrašnja duševna stanja bića, razmišljanja o sebi, o problemu, opreznost.

Tradicionalno učenje favorizuje verbalno-lingvističku i logičko-matematičku inteligenciju. Gardner sugerise izbalansirane programe koji uključuju umetnost, komunikaciju, samoaktuelizaciju i fizičku edukaciju. Svakom pojedincu treba omogućiti razvoj svih inteligencija, uključujući glumu, muziku, kooperativno učenje, razmišljanje, vizuelizaciju, naraciju. Ocenjivanjem treba obuhvatiti sve vrste inteligencija kao i samotestirajuće metode koje pomažu studentima da spoznaju svoje sklonosti (dominantne inteligencije).

Danas se intenzivno razmatra način korišćenja teorije višestruke inteligencije radi personalizacije učenja uz podršku tehnologije [1]. Jedan od pristupa personalizacije je da se prepozna tip inteligencije koji je najvažniji da se nastavni sadržaj usvoji, i na osnovu toga da se odabere odgovarajuća tehnologija predstavljanja istog. Sledeća ilustracija prikazuje način mapiranja tipova inteligencije, vrste sadržaja (načine objašnjavanja) i odgovarajuće tehnologije za prezentaciju sadržaja (Tabela 4). Polazište ovakvog pristupa je u činjenici da svi ljudi poseduju sve tipove inteligencije (u različitim proporcijama).

Tabela 4: Veza tipa inteligencije i tipa tehnologije koja se može primeniti na proces učenja (Dara-Abrams 2002)

Intelligence	Explanation Variant	Technology
Linguistic	Prose, Textual Explanation	HTML, Word
Logic-Math	Bulleted List	HTML list
Spatial	Diagrams, Graphics, Movies	Flash, iMovie, Power Point
Musical	Sound Effects, Sound Track	Flash, Audio
Intrapersonal	Self-Guided Problem Analysis, Journals	HTML forms with script
Interpersonal	Discussions – problems, cases, questions	Threaded discussion
Naturalist	Categories and Metaphors	HTML lists, Flash
Bodily- Kinesthetic	Hands-on Exercises Simulations	Scripts Virtual Environments

Osobe sa izraženom lingvističkom inteligencijom se dobro snalaze sa tekstualnim sadržajima; logičko-matematička inteligencija omogućava brzo prepoznavanje obrazaca, i bolje snalaženje sa sažetim struktuiranim podacima predstavljenim u vidu lista. Prostorna inteligencija nam omogućava da učimo iz vizuelnih reprezentacija informacija, a muzička inteligencija iz zvučnih zapisa. Telesno-kinetička inteligencija omogućava koordinaciju misli i pokreta u skladu sa situacijom u kojoj se osoba nalazi, bilo da se radi o simuliranim ili stvarnim okruženjima za učenje raznih veština.

2.1.8 Teorija kontrole

Teorija kontrole zasnovana je na motivaciji studenata (William Glasser, 1990.) [57]. Za razliku od biheviorističkog favorizovanja uticaja okruženja na pojedinca, Glasser smatra da ponašanje nije odgovor na spoljni stimulus, već je određeno motivima (željama): opstanak, ljubav, moć, snaga, sloboda i druge osnovne ljudske potrebe. Na primer, pojavu nemotivisanosti studenata za nastavu objašnjava mišljenjem studenata da je ta nastava nesvrhsishodna za njihove potrebe.

Korišćenje nagrade i kazne radi prisiljavanja studenata da pristanu na pravila iz nastavnog procesa (u vezi učenja i testiranja) Glasser naziva učenjem na mišiće, a nastavnike koji koriste prinudu - šefovima. Studenti u takvoj situaciji svoj rad tretiraju kao otaljavanje, čak i kada su nagrađeni.

Nastavnike koji izbegavaju ucenjivačke metode, koriste ocene kao privremene pokazatelje znanja studenata, koji primenjuju suštinske nagrade zauzimajući jasan stav prema studentima i koji koreliraju ocenjivanje sa studentovim osnovnim potrebama, Glasser naziva vođama. Smatra da ovaj tip nastavnika jako štiti studente koji se ističu u motivisanosti, predanom radu i aktivnosti.

Nastavnici posreduju između sadržaja i metoda nastave na jednoj, i studenata na drugoj strani. Studentima je potrebno jasno ograničiti šta i koliko treba da uče. U nastavi treba primenjivati tehnike aktivnog i kooperativnog učenja, koje uvećavaju mogućnosti studenata. Ocenjivanje treba da bude povezano sa određenom studentskom satisfakcijom – da vrednuju kvalitetan rad i razvijaju potrebu za usavršavanjem.

2.1.9 Neurologija

Neurologija, posmatrana kao teorija učenja, je naučna studija o ljudskom nervnom sistemu, mozgu i biološkoj osnovi percepcije, učenja, pamćenja i svesti [57]. Neurologija povezuje posmatranja kognitivnog ponašanja sa aktuelnim fizičkim procesima koji podržavaju ta ponašanja. Ključni pronalasci neurologije su više egzaktna naučna otkrića iz domena medicine. Iz ugla teorije učenja najznačajnija su da postoje odvojena područja korteksa za različite mentalne procese, da je mozak samoorganizujući sistem koji za šta gde treba uskladištiti i da je ovaj sistem promenljiv tokom života - veze između pojedinih zona se ojačavaju ponavljanjima (memorija i mišljenje).

Kada nastavnici uzmu u obzir neurologiju, oni organizuju nastavne programe oko realnog iskustva i integrišu čitave ideje. Čak se fokusiraju na instrukcijama koje promovišu kompleksno razmišljanje i razvoj mozga. Neurologija protežira kontinuirano učenje i intelektualni razvoji u zreloom dobu.

2.1.10 Teorija učenja fokusirana na ulogu mozga (Brain-based Learning)

Slično neurologiji, ova teorija je bazirana na strukturi i funkciji mozga. Dok god mozak normalno funkcioniše, učenje je uspešno [57]. Ljudi često kažu da svako može da uči. Ipak, realnost je da svako uči! Svako je rođen sa mozgom čije su funkcija kao ogroman moćan procesor. Tradicionalno školovanje guši učenje obeshrabrivanjem, ignorisanjem,

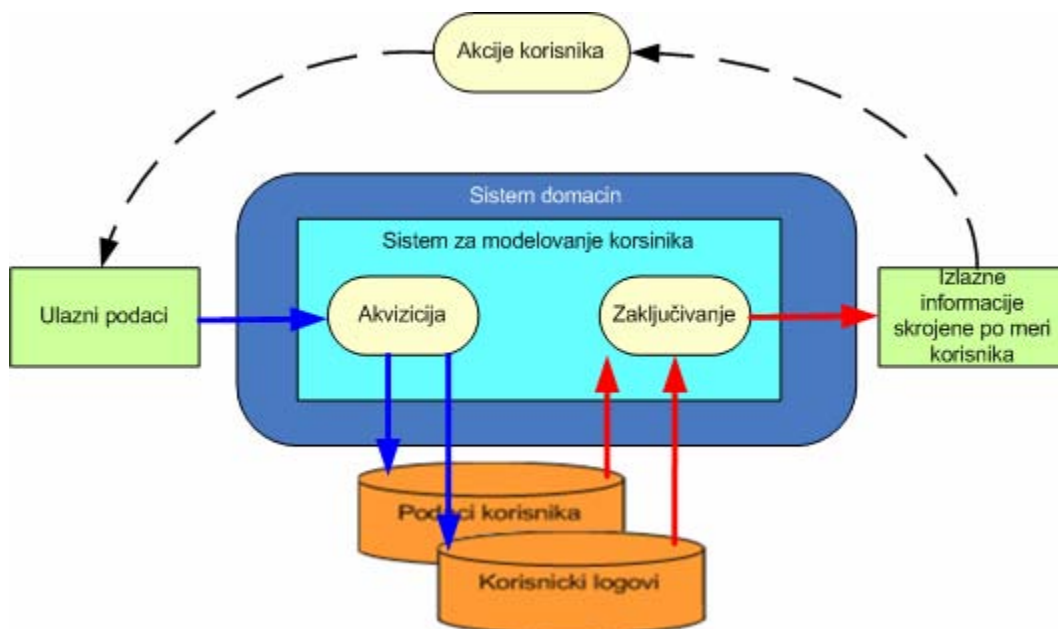
kažnjavanjem moždanog prirodnog procesa učenja. Ova teorija protežira tri tehnike učenja:

1. Kreiranje okruženja za učenje koje potpuno utapa studente u jedno edukaciono iskustvo (*Orchestrated immersion*)
2. Pokušavanje da se eliminiše grozničavo učenje, održavanjem okruženja sa puno izazova (*Relaxed alertness*)
3. Obezbeđivanje učenika da konsoliduje i internalizuje informacije kroz aktivnu obradu istih (*Active processing*)

Ova teorija zagovara stav da nastavnici moraju dizajnirati učenje oko studentovih interesovanja i učiniti učenje kontekstualnim. Nastavnici da animiraju studente da uče u grupama. Učenje treba da se fokusira oko realnih problema, a studenti da budu motivisani za učenje što više van škole i učionice. Ova teorija je značajna jer pokušava da spozna koje vrste aktivnosti učenja su najefektivnije. Dizajneri edukacionih alati moraju biti umetnici u kreaciji okruženja za učenje.

2.2. Modelovanje korisnika

Svi informacioni sistemi uopšte, sadrže implicitne ili eksplicitne korisničke modele. Ovi modeli se u literaturi nazivaju različitim terminima: korisnički modeli, profili, podaci, nalozi, korisničke dozvole i uloge.



Ilustracija 2: Princip rada sistema za modelovanje korisnika

Zajedničko za sve korisničke modele je da su namenjeni pohranjivanju podataka individualnih korisnika, relevantnih za posmatrani informacioni sistem. Na osnovu korisničkih modela, sistem omogućava da se na odgovarajući način koriste ostali softverski i hardverski resursi (Ilustracija 3).

Modelovanje korisnika implementirano je u okviru posebnog sistema, koji može biti deo informacionog sistema (u daljem tekstu IS) domaćina, ili nezavisna softverska

komponenta. U drugom slučaju IS domaćin ovu komponentu koristi preko javno dostupnih API (*Application Programming Interface*) funkcija (jaka sprega), ili preko javnih servisa (slaba sprega).

Ulazni podaci korisnika su svi oni podaci koje sistem prikuplja od korisnika pri njegovom registrowanju na sistem (pri otvaranju naloga, popunjavanju različitih vrsta upitnika), i u toku korišćenja sistema. Jednom rečju sistem vrši akviziciju podataka korisnika, pohranjujući ih u neko skladište (u bazi podataka ili fajl sistemu). Na osnovu prikupljenih podataka, po zahtevu sistema domaćina, sistem za modelovanje donosi potrebne zaključke o korisniku, koje sistem domaćin koristi radi adaptacije sadržaja za korisnika. Celokupan proces se odvoja dok god traje sesija korisnika sa sistemom, tako da svaka sledeća korisnička akcija generiše nove podatke za sistem, koji se nanovo akviziraju, skladište i o njima se vrši ponovno zaključivanje. Sistem za modelovanje korisnika, međutim nije prost sistem sa povratnom spregom iz dva razloga: pored aktuelnih, koriste se i istorijski podaci korisnika i drugi, proces akvizicije podataka i proces zaključivanja o korisniku su potpuno asinhroni i izvode se isključivo po zahtevu sistema domaćina.

Studentski model predstavlja pod-tip korisničkog modela, koji ga proširuje i koji je implementirani u edukativnim sistemima.

2.2.1 Uopšte o korisničkim modelima

Potreba modelovanja korisnika uočena je još početkom '70-ih. Najranija istraživanja u modelovanju korisnika vršena su u kontekstu istraživanja sistema koji su omogućavali dijalog sa korisnikom prirodnim jezikom. Kobsa, Koenemann, i Pohl [13] su predstavili teoretski jednu od najpotpunijih specifikacija korisničkog modela – koje podatke o korisniku sistem treba da prikuplja i da čuva u cilju personalizacije servisa i sadržaja koje sistem nudi. Korisnički podaci mogu se grupisati u dve velike celine (Tabela 5). Podaci koji opisuju mentalno stanje korisnika (*podaci o korisniku*) i podaci o korisnikovim akcijama. Prva grupa podataka ima statičku prirodu (preciznije, manje su podložni promenama) i predstavljaju personalne karakteristike korisnika. Ovi podaci su generalni – njima se opisuje bilo koji tip korisnika u bilo kom tipu sistema. *Podaci o korisniku* su potpuno nezavisni od sistema i može se slobodno reći da predstavljaju korisnikov portfolio, sastavljen na eksplicitan način (popunjavanjem formulara, upitnika i anketa). Ovi podaci su najčešće i transparentni za različite delove sistema koji ih koristi, pa čak i za spoljne korisnike (druge sisteme).

Tabela 5: Korisnički model (Kobsa, Koenemann, i Pohl, 2001)

KORISNIČKI MODEL		
Podaci o korisniku	Podaci o korišćenju sistema	
	Opservirano korišćenje	Regularnost korišćenja
• Demografski • Znanje • Veštine i sposobnosti • Tehnološka iskustva	• Selektivne akcije • Povremeno posmatrana ponašanja • Merenja • Ostale	• Frekvencija korišćenja • Korelacija situacija – akcija • Redosled akcija

• Individualne crte • Interesovanja i preference • Ciljevi i planovi	konfirmatorne akcije	
--	----------------------	--

Demografski podaci obuhvataju osnovne generalije kao što su ime i prezime, adresa stanovanja, region, država, pol, obrazovanje.

Znanjem je predstavljeno koliko korisnik poznavanje pojedine koncepte kojima se sistem bavi. *Veštine i sposobnosti* su važne radi procene koliko će korisnik biti sposoban za korišćenje sistema.

Interesovanja i preference su veoma značajne za personalizaciju servisa koji sistem nudi korisniku, jer jedan od osnovnih ciljeva personalizacije je da privuče korisnika da koristi sistem (karakteristično za takozvane *recommender-e* – sisteme za preporučivanje).

Ciljevi i planovi korisnika mogu biti vrlo problematični. Korisnik može da koristi sistem bez jasnog plana ili cilja. Ovi podaci su vrlo važni za edukativne sisteme, jer je potrebno da korisnik (student) shvati i usvoji zadate obrazovne ciljeve i planove njihove realizacije.

Model takođe podržava podatke koji omogućavaju praćenje ponašanja korisnika u toku eksploatacije sistema. Ti podaci mogu biti direktno posmatrani i zapisani, ili se do njih može doći na osnovu opserviranih podataka. Postoji veliki broj tipova interakcija korisnika sa sistemom.

Selektivne akcije predstavljaju najčešće korišćene akcije, koje mogu poslužiti za prepoznavanje korisnikovih interesovanja. Na primer selekcijom linkova korisnik zapravo vrši navigaciju kroz informacioni prostor aplikacije. Sistem na osnovu ostvarenih putanja može da donese zaključak o interesovanjima i motivima korišćenja sistema. Selektivne akcije mogu da omoguće prepoznavanje neznanja konceptata, naročito u tehnološkom smislu (aspekti učenja i korišćenja sistema). Prepoznavanje preferenci je treća prednost praćenja selektivnih akcija. Izborom proizvoda koji se razlikuju u dizajnu, umetničkih dela, eksponata, fotografija, specifičnih grafičkih okruženja, sistem može da prepozna estetska merila, stil i ukus krajnjeg korisnika.

Povremeno posmatrano ponašanje predstavlja način da sistem na osnovu merenja vremena korisnikovog zadržavanja na stranici proceni njegovu zainteresovanost prezentovanim materijalom. Ovo merenje je diskutabilno jer aplikacija koja je serverska nije u mogućnosti da registruje sve merodavne akcije korisnika. Iz tog razloga uvodi se klijentska aplikacija (deo sistema koji se izvršava na korisničkom računaru), koja treba da snima ponašanje korisnika i šalje podatke o tome serverskoj. Na primer, da li prozor aplikacije ima fokus, da li je minimiziran, da li je korisnik ispred računara i druga slična ponašanja. Vrlo je teško implementirati ovakav način opserviranja korisnika (zahteva saglasnost korisnika, sinhronizaciju sa serverskom aplikacijom, intenzivnu interakciju između klijentske i serverske strane, instaliranje komponente na klijentskom računaru i sl.), i najčešće odvraća korisnika od korišćenja sistema (ugrožavanje privatnosti, frustriranost ograničavanjem akcija). Zbog toga podaci prikupljeni na ovaj način imaju vrlo negativnu konotaciju u edukativnim sistemima, izuzev kada su neophodni (na primer, kontrolisanje ponašanja studenata u toku proveravanja).

Merenja se koriste u nekim sistemima, radi omogućavanja korisnicima da ocenjuju sistemске resurse (dokumenta, proizvode, nove artikle i sl.). Za edukativne sisteme, ovaj mehanizam je vrlo popularan u procesima evaluacije sistema, nastavnih materijala, predavanja, nastavnika.

Pored navedenih (merenja, ocenjivanja, potvrda izbora) postoje i druge konfirmatorne akcije (slanje poruka, elektronske pošte, učešće na forumima i diskusionim grupama, upis beleški, privlačenje – *download* materijala i sl.), koje mogu biti predmet beleženja i merenja radi profilisanja korisnika.

Pored direktnih podataka, sistem može da izvede zaključke o korisniku i indirektno, na osnovu statistički obrađenih podataka. Kategorizacija događaja i merenje njihove *frekventnosti* verovatno predstavlja jedan od najčešće korišćenih načina procesiranja. Merenjem frekventnosti dobijaju se pouzdani podaci o interesovanjima i preferencama korisnika.

Korelacija situacija – akcija otkriva uzročno – posledične veze u ponašanju korisnika. Različiti paramteri se mogu meriti na ovaj način: korisnikov stil rada (učenja), stepen motivacije pri korišćenju sistema, spretnost i veština u korišćenju sistema, nivo znanja korisnika u oblasti od interesa.

Redosled akcija se najčešće analizira radi preporuka i optimizovanja navigacije u informacionom prostoru, radi predikcije budućih akcija korisnika na osnovu redosleda proteklih, i radi preporuka akcija na osnovu redosleda akcija drugih korisnika.

2.2.2 Tipologizacija korisničkih modela

Postoje različite klasifikovanje korisničkih modela [18]. Osnovna klasifikacija, koja je ujedno i najstarija korisničke modele svrstava u dve grupe: prekrivajući i stereotipni.

2.2.2.1 Prekrivajući korisnički model

Jednostavnost prekrivajućeg modela je u njegovoj definiciji: on predstavlja znanje korisnika u odnosu na sveukupno društveno znanje. Model se dakle implementira tako da se znanje korisnika predstavlja podskupom znanja koje je na neki način formalizovano (pohranjeno u vidu skupa činjenica, i/ili implementirano kroz pravila, ili ograničenja) u sistemu (Ilustracija 4).

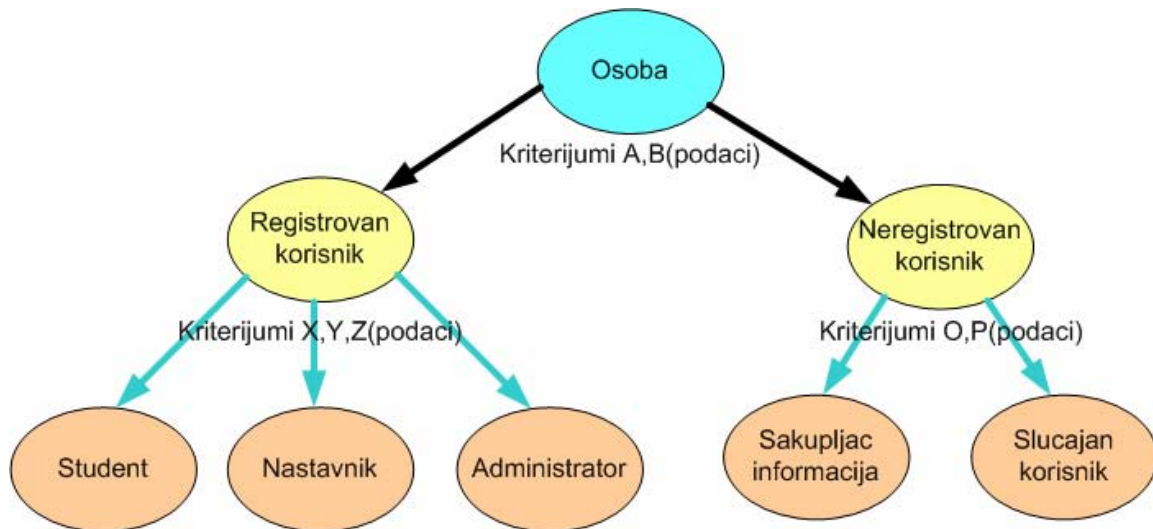


Ilustracija 3: Prekrivajući (overlay) model

Ovaj model se često naziva i Sokratovski model, jer je ograničen znanjem koje društvo uopšte poseduje iz ciljnog domena (ciljne naučne oblasti). Pošto je znanje proširljivo, tako nema ni ograničenja u prekrivajućem modelu. Ekspertsko znanje se koristi radi predstavljanja opšteg društvenog znanja. Glavni problem ovog modela je inicijacija korisničkog modela. Korišćenjem prekrivajućeg modela, sistem modeluje samo ono što korisnik zna, ali ne i greške u razumevanju koncepata. Uprkos navedenim slabostima, prekrivajući model je aktuelan i koristi se u edukativnim aplikacijama: ELM-ART [15], KN-AHS [16], ORIMUHS [17].

2.2.2.2 Stereotipni model

Sušтина stereotipnog modela je da se korisnik na osnovu ulaznih podataka svrsta u neki od unapred definisanih stereotipa. Stereotipi predstavljaju kategorije korisnika i podrazumeva se da korisnici koji su opisani istim stereotipom, imaju iste (slične) karakteristike, koje kroz implementaciju nasleđuju od stereotipa (Ilustracija 5).



Ilustracija 4: Stereotipni model

Ovaj model je naročito zastupljen u *recommender*-ima – sistemima koji generišu razne vrste preporuka korisnicima sistema. Prednost stereotipnog modela je da sistem na osnovu male količine informacija može da klasifikuje konkretnog korisnika. Zahvaljujući hijerarhijskoj karakteristici, stereotipni model je kompatibilan sa stablom odlučivanja (klasifikovanja). Za svaku odluku se vezuje po jedan stereotipu, pri čemu je dozvoljeno nasleđivanje karakteristika od roditeljskih stereotipova. U primeru sa prethodne ilustracije može se zaključiti da ako ne postoji nikakav podatak o korisniku, on se svrstava u najopštiji stereotip (*Osoba*). U slučaju da se model dopuni podacima (*A* i *B*), korisnik se ponovo razvrstava u jedan od stereotipova (*Registrovani/neregistrovani korisnik*). U slučaju da se model ponovo dopuni podacima (*X,Y,Z* ili *O,P*), i razvrstavanje se ponavlja.

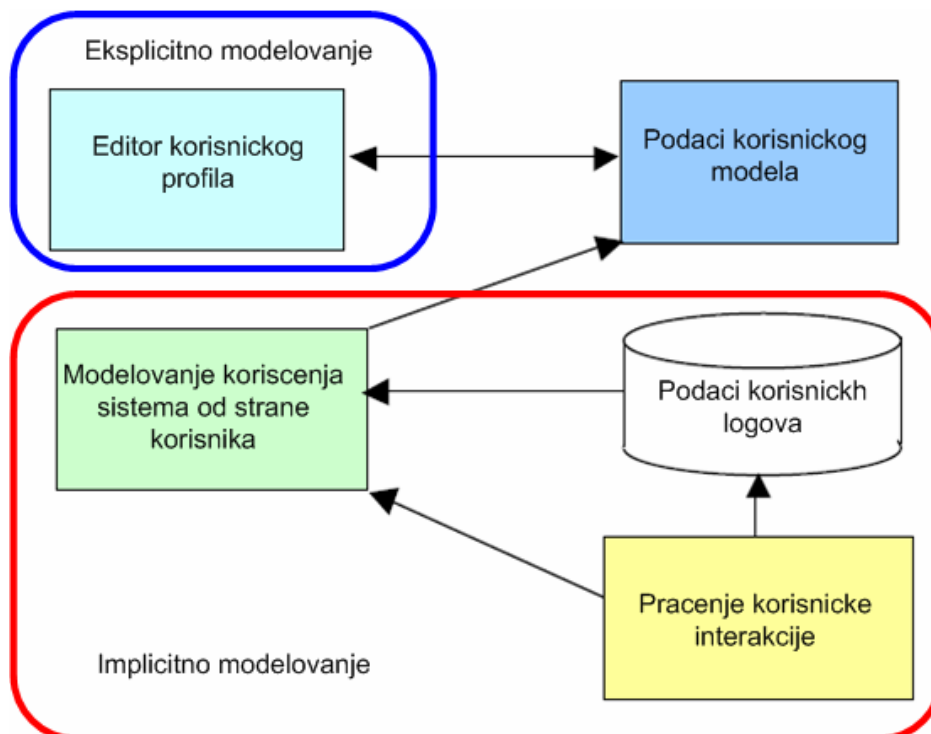
Iz navedenog opisa, može se uočiti i druga značajna karakteristika: osnovu stereotipnog pripadanja, sistem može da izvrši predikciju šta korisnik želi, buduće akcije (ponašanje). Ugrađivanjem mehanizama povratne sprege (obrada korisnikovih reakcija, zaključivanje

i korekcija kriterijuma klasifikacije) ovaj model je proširljiv i pogodan je za dinamičke modifikacije.

2.2.2.3 Implicitni i eksplicitni modeli

Da bi se bilo kakva aktivnost korisnika personalizovala, potrebno je postojanje korisničkog modela. Međutim postoje sistemi (npr. sajt www.amazon.com) koji vrše personalizaciju bez potrebe registrovanja i logovanja korisnika. Personalizacija se vrši na osnovu praćenja ponašanja (akcija) korisnika i uklapanju korisnika u neki od poznatih obrazaca (stereotipa). Na taj način sadržaji, koji se dostavljaju od strane sistema, su zaista personalizovani (najčešće su to preporuke i saveti, različita postavljanja navigacije, pravljenje prečica). Iako ne postoji eksplicitni model korisnika, zahvaljujući generalnom pristupu korišćenja obrazaca ponašanja, podaci korisnika se evidentno opserviraju i na osnovu njih se donose zaključci koji se koriste za različite tipove adaptacije.

Za implicitne modele (Ilustracija 6) karakteristično je da se podaci za istog korisnika čuvaju distribuirano i obrađuju parcijalno u različitim komponentama sistema. Kriterijum njihove integracije nije vezan za konkretnu osobu. Korisnički podaci se najčešće sublimiraju kroz statističku obradu radi dobijanja podataka vezanih za eksploataciju samog sistema (na primer analiza posećenost sistema u vremenskom domenu, korelacija aktivnosti korisnika sa drugim aktivnostima u sistemu ili okruženju, analiza tipova korisnika). Sistemi sa implicitnim korisničkim modelom koji zahtevaju registrovanje korisnika imaju problema sa očuvanjem integriteta i konzistentnosti korisničkih podataka. Korisnici se mogu višestruko registrovati u sistemu (na primer, usled zaboravljanja podataka korisničkih naloga), što dovodi do pojave redundantnih podataka.



Ilustracija 5: Implicitno i eksplicitno modelovanje

Za razliku od implicitnih modela, eksplicitni korisnički model predstavlja integralnu predstavu realnog korisnika u sistemu. Za eksplicitni model je potrebno registrovanje korisnika radi prikupljanja neophodnih podataka za rad sistema. Podaci koji se pohranjuju u ovakvim modelima imaju perzistentnu prirodu – sistem ih čuva duži vremenski period. Pošto je stepen personalizacije servisa, koji sistem nudi korisniku, u direktnoj srazmeri sa kompleksnošću korisničkog modela, poželjno je da model podržava što više podataka o korisniku (što više podataka o korisniku, veće mogućnosti personalizacije).

Edukativne aplikacije, bez obzira da li poseduju mogućnost personalizacije sadržaja i servisa koje sistem nudi korisniku, poseduju isključivo eksplicitne korisničke modele. Potreba studenta da ima uvid u rezultate svog rada, pregledi rezultata i napredovanja studenata u toku izvođenja nastave za nastavnike, primena različitih pedagoških metoda i strategija, predstavljaju neke od zahteva koji iziskuju postojanje eksplicitnog korisničkog (studentskog) modela. Iz ugla administriranja, neuporedivo je lakše održavanje integriteta i konzistentnosti korisničkih podataka kod eksplicitnog modela. Eksplicitni korisnički model takođe može biti distribuiran radi postizanja portabilnosti (dinamičko uključivanje pojedinih delova modela prema potrebi sistema), međutim i pored toga ne postoji suvišna redundansa podataka.

Korisnički modeli predstavljaju osnovu za modelovanje studenata u edukativnim procesima. U praksi, razvijeni su međunarodni standardi – studentski modeli, kojima je omogućena razmena podataka o studentima između različitih sistema (edukativnih i poslovnih), personalizacija učenja, formiranje ličnog konzistentnog portfolija, primena koncepta doživotnog učenja, saradnja između svih edukativnih subjekata (studenata, nastavnika, edukativnih ustanova, eksperata, instituta i sl.), i još mnogo drugih prednosti. Konkretna specifikacije studentskih modela predstavljene su u sledećem poglavlju rada.

3. Analiza

U ovom poglavlju detaljno su predstavljene aktuelni standardi za modelovanje podataka studenata i sadržaja učenja, a koji predstavljaju osnov za personalizaciju procesa učenja. U drugom delu su opisani različiti tipovi edukativnih sistema koji se primenjuju školama i na fakultetima, uz isticanje prednosti i nedostataka u eksploataciji. Za svaki od tipova predstavljene su reprezentativne implementacije (konkretna softverska rešenja).

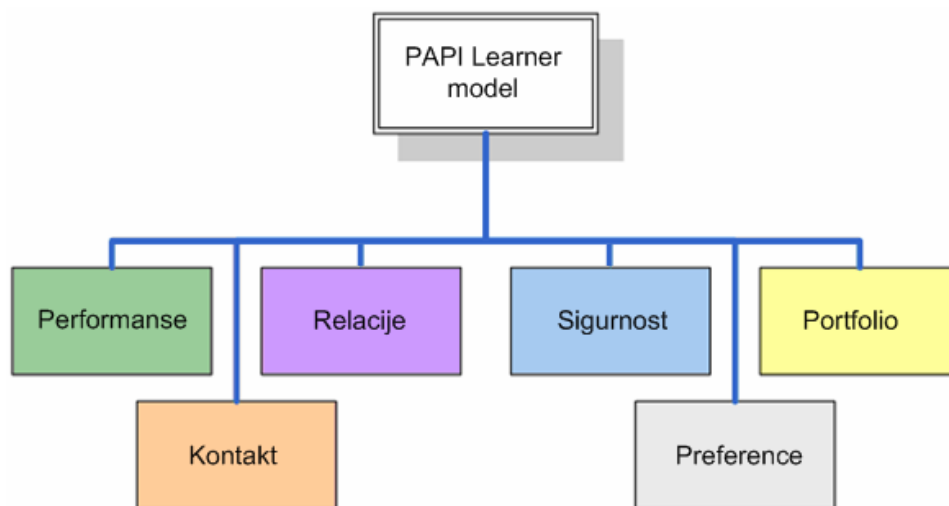
3.1 Studentski modeli

Studentski model predstavlja jednu od najvažnijih komponenti edukativnih sistema (aplikacija) koji omogućavaju personalizaciju sadržaja učenja. Model, zavisno od potreba konkretne aplikacije kojoj je namenjen, može da sadrži informacije o znanju, veštinama, sposobnostima, preferencama ili nekim drugim bitnim karakteristikama studenta. Model studenta treba da omogući predstavljanje svih relevantnih informacija o studentima u edukativnim aplikacijama. učenika. deo sistema Postoji različiti standardi za modelovanje studenata od kojih se mogu izdvojiti IEEE PAPI Learner, IMS LIP, EduPerson, ELENA project LM i FOAF.

3.1.1 IEEE PAPI (Public and Private Information) Learner

Ovaj standard (IEEE P1484.2.21/D8) [19] je preporučen od strane ISO/IEC JTC (Joint Technical Committee) WG2 (Workgroup 2), međunarodne organizacije za propisivanje standarda u informacionim tehnologijama [20]. PAPI model je namenjen za opisivanje osnovnih informacija o studentima, kao i za razmenu tih informacija između sistema koji se nalaze u međusobnoj interakciji. PAPI predstavlja standard sastavljen iz više delova koji specificiraju semantiku i sintaksu informacija o studentu (Ilustracija 7).

Osnovna ideja ovog modela je da se formiraju mali portabilni informacioni moduli koji mogu da se koriste u aplikacijama potpuno odvojeno. U odnosu na prethodno opisani korisnički model (Kobsa, Koenemann, i Pohl), PAPI model ističe socijalnu komponentu (povezanost, relacije, kontakte) i bezbednosnu politiku (različita dostupnost i nivoi zaštita tajnosti korisnikovih podataka).



Ilustracija 6: IEEE PAPI Learner – standardni studentski model (IEEE P1484.2/D7)

Ceo skup pomenutih informacija čini *Studentski profil* – profil individualnog studenta. Informacije o studentu mogu biti kreirane, skladištene, izmenjene, korišćene od strane samog edukativnog sistema, pojedinaca (bilo da se radi o studentu ili nastavniku), ili drugih entiteta (drugi povezani sistemi, ili pojedinci, na primer administratori, osoblje studentskih službi). PAPI koristi više različitih tipova informacija o studentu – *informacionih modula*. Oni uključuju informacije o kontaktima, preferencama, performansama, lični portfolio (zvanja, licence, sertifikati), sposobnostima i druge tipove informacija. PAPI standard, obuhvata šest informacionih modula. Za svaki od njih PAPI standard opisuje podskup koji je koristan i koji se može implementirati na različite načine. Ovaj standard je otvoren za proširenja, a uključuje minimum informacija koji je neophodan da zadovolji zahteve maksimalne prenosivosti. Sledi opis pojedinih informacionih modula.

- *Kontakt* – ovaj modul informacija obuhvata lične podatke (ime, prezime, jedinstveni identifikacioni broj), ali i informacije potrebne za ostvarivanje kontakta (poštanske adrese, telefonski brojevi, faksimil brojevi, e-mail adrese). Nije direktno povezan sa merenjem i snimanjem performansi studenta, a primarno

je povezan sa administracijom. Normalno, ovaj tip informacija treba da bude privatn i bezbedan, dostupan samo entitetima koji su za to ovlašćeni.

- *Performanse* – informacije koje su tesno povezane za edukativne procese. To su podaci o sadržaju na koji se odnosi performansa, izmerenu vrednost performanse (ocena), korišćenu metriku ocenjivanja, tačno vreme i datum proveravanja, period važenja ocene, identifikatore studenta i lica koje je unelo podatke o performansi, lista sertifikata, granularnost (veličina) proverenog sadržaja i kompetencije na osnovu performanse.
- *Relacije* – relacije opisuju povezanost studenta sa drugim korisnicima sistema za učenje, bilo da se radi o profesorima, administratorima ili drugim studentima. Ove informacije uključuju tipove relacija (*predaje, podučavan_od, pripada, druži_se_sa*, itd), identifikatore drugih osobe sa kojima je student u relacijama i liste uređenih parova osoba – relacija.
- *Bezbednost* – ovaj aspekt je novitet u modelovanju studenta, i predstavlja deo implementacije administrativne funkcionalnosti. Bezbednosni modul omogućava korisniku da poseduje različite naloge za različite kontekste korišćenja sistema (s posla, od kuće, kao student, kao instruktor, kao administrator, itd.). U ovom modulu se mogu naći pored korisničkih imena i lozinki, vrednosti izazova/odgovora¹, privatni i javni ključ, biometrijski podaci i slično.
- *Preference* – ovaj informacioni modul je fokusiran na adaptibilnost u interakciji između korisnika i sistema. Predstavlja skup informacija o studentu - preference pri korišćenju sistema. Na primer: karakteristike teksta (veličina teksta i font), boja slova i pozadine, preferirane vrste kontrola na ekranskim formama (izbor iz padajuće liste ili izbor pomoću radio dugmadi), količina informacija na jednoj stranici (broj pasusa, ili fragmentiranost teksta), prikazivanje ilustracija u posebnim prozorima ili u samom tekstu, redosled gradiva (prvo teorijski koncepti, ili prvo primer). Pored navedenih postoje mnoge druge vrste preferenci koje se koriste, ili ne u zavisnosti od namene i kompletnosti implementacije sistema.
- *Portfolio* – studentski portfolio predstavlja reprezentativnu kolekciju rada i referenci studenta. Ova kolekcija, namenjena za ilustraciju i obrazlaganje studentovih sposobnosti i dostignuća, obuhvata podatke stručne biografije i deo vezan za produkciju publikacija i postignute rezultate. Modul portfolija je vrlo značajan za socijalni kontekst učenja. On omogućava potpunije sagledavanje studenta, predikciju budućih rezultata i efikasnije ostvarivanje relacija između studenata, nastavnika u kontekstima učenja i zajedničkih interesovanja.

Postoji više razloga zbog kojih se može reći da je *PAPI Learner* model najznačajniji standard za modelovanje studenata. Preporučen je od referentne međunarodne organizacije (IEEE) i neprekidno se proširuje i usavršava (trenutno aktuelna je verzija 8 modela). Model ima izraženu skalabilnost: nezavisnost informacionih modula omogućava njihovo selektivno korišćenje – prema potrebama konkretnog sistema. Ne

¹ Vrednosti izazova/odgovora predstavljaju vrednosti naophodne za autentikaciju u sistemima zaštite tajnosti podataka.

moraju svi moduli da budu uključeni u implementirano rešenje. Pri tome model nastavlja da bude interoperabilan sa drugim sistemima koji imaju podršku za *PAPI Learner* – a.

Ovaj model poseduje i veliku fleksibilnost. Za svaki informacioni modul dozvoljena su proširenja, s tim da se ne narušavaju postojeći koncepti standarda. U slučaju da je potrebno postići interoperabilnost sa studentskim modelima, koji su nekompatibilni sa *PAPI Learner* modelom, standardom je predviđeno da se u tom slučaju ne prave ekstenzije postojećih modula, već da se povezivanje obavi preko novih kreiranih modula. Slabom spregom između novonastalih i postojećih modula i ekstenzijama koje ne prevazilaze veličinu standardnih jezgara modula, očuvana je kompaktnost modela.

Pored navedenih prednosti, ovaj model ima i nekoliko nedostataka. *PAPI Learner* model nema kategoriju ciljeva [21]. U kontekstima učenja, ciljevi su izuzetno značajni iz aspekta praćenja napredovanja studenta. Studenti imaju različite ciljeve (na primer dostizanje različitih rezultata, nivoa znanja i sl.). Da bi edukativni sistem na pravi način odgovorio zahtevima konkretnog studenta (da bude agilan u vremenu i kontekstu), potrebno je da stalno meri razliku ostvarenih rezultata i planiranih ciljeva. U praksi postoje edukativni sistemi čija se adaptibilnost zasniva isključivo na merenju ostvarenosti konkretnih ciljeva učenja (na primer inteligentni tutorski sistemi kod kojih je implementirano učenje rešavanjem problema). Drugi nedostatak *PAPI Learner* modela je nepostojanje modula za praćenje akcija studenata tokom korišćenja sistema. Ovaj, takozvani *tracking* modul, je važan radi donošenja detaljnijih zaključaka o ponašanju studenata tokom korisničkih sesija i vrlo često je implementiran u sistemima u praksi (detaljnije u poglavlju *Analiza*).

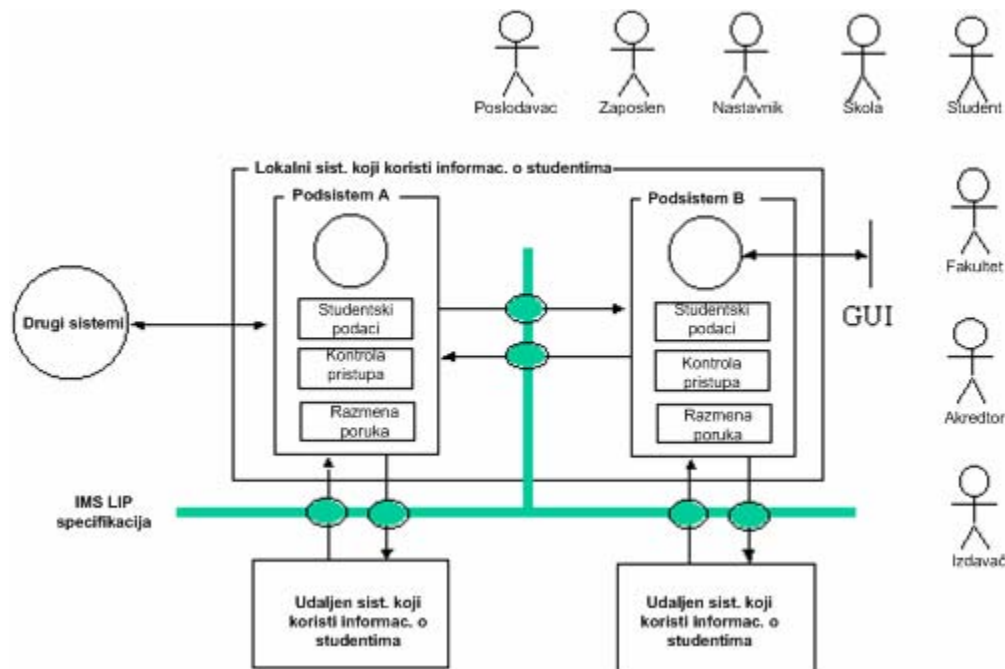
3.1.2 IMS LIP (Learner Information Package)

IMS (Instructional Management Systems) je neprofitna organizacija koja obuhvata preko 50 različitih kompanija (koje se bave razvojem edukativnih softverskih proizvoda) radi postizanja uzajamne interoperabilnosti i zajedničkog nastupa na IT tržištu. *IMS* propisuje interfejs za povezivanje elektronskih sadržaja za učenje (pre svega nastavni materijali i testovi). Pored navedenog, *IMS* konzorcijum dizajnirao je *LIP (Learner Information Package* – korisnički informativni paket) s sledećim motivima:

- *Distribucija informacija* – podaci studenta mogu biti distribuirani u različitim delovima sistema. Mogu biti redundantni. Iz tog razloga model mora da ima izraženu vremensku dimenziju i mogućnost održavanja konzistentnosti distribuiranih informacija (kroz mehanizme indeksiranja i vremenskih logova). Standard treba da, samim dizajnom pakovanja podataka, učini dostupnim informacije o studentu za sve delove sistema, ili različite sisteme koji kooperiraju.
- *Skalabilnost* – radi podrške različitim tipovima edukativnih sistema, standard treba da omogući razmenu i repakovanje informacija o studentu. U tom cilju, informacije o studentu treba da imaju proizvoljnu granularnost, a u razmeni je moguća pojava transfera čitavih paketa, ili samo njihovih delova. Da bi se to omogućilo, studentski model može da se sastoji od većeg broja manjih informacionih paketa (više *IMS LIP* – ova), pri čemu se u kompoziciji koriste *IMS Content&Packaging* specifikacija.

- *Privatnost i zaštita podataka* – edukativni sistem treba da bude sposoban da implementira različite mere privatnosti, zaštite podataka i da osigura njihov integritet.
- *Fleksibilnost i mogućnost referenciranja spolja* – informacije o studentu su najčešće izvedene, i kao takve – struktuirane i kompleksne. Na primer ciljevi i istorija učenja se različito interpretiraju u različitim kontekstima, za potrebe različitih delova sistema ili za druge edukativne sisteme. Model podataka mora da bude dovoljno fleksibilan da bi se to omogućilo.

U razvoju *IMS* – ove specifikacije dizajniran je generalni model sistema [22] koji bi eksploatisao *IMS LIP* studentski model (Ilustracija 8).



Ilustracija 7: Model sistema koji bi eksploatisao IMS LIP model studenta

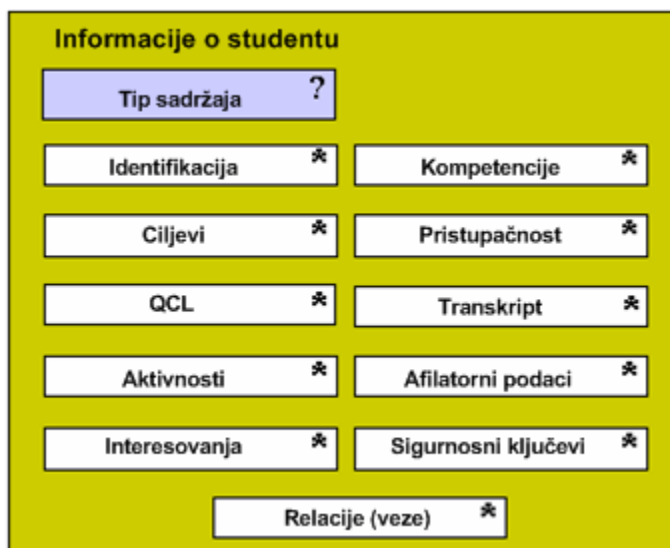
U sistemskom modelu projektovano je osam korisničkih uloga od kojih su šest fizička lica (*poslodavac*, *zaposlen*, *nastavnik*, *student*, *akreditor* i *izdavač*) i dve institucije (*škola* – za osnovno i srednje obrazovanje, i *fakultet* – za visokoškolsko obrazovanje). Pretpostavka je da se podaci o studentima čuvaju na različite načine u različitim delovima sistema (podsystemi A i B). Pored studentskih podataka, postoje podaci za kontrolu pristupa (zaštita privatnosti podataka) pojedinih korisnika i korisničkih uloga, i istorijski podaci – poruke koje podsystem razmenjuje sa okruženjem. Razmena informacija između entiteta u sistemu se vrši isključivo posredstvom poruka u LIP formatu (IMS LIP specifikacija). I drugi sistemi koji ne podržavaju ovaj standard, takođe mogu razmenjivati informacije o korisnicima, ali je potrebno izvršiti mapiranje² sa IMS LIP specifikacijom.

² Mapiranje standarda predstavlja meta-meta modelovanje, koje takođe uključuje posebne tehnologije i ontološke teorije (npr. korišćenje OWL-a, ili DAML-OIL jezika za opis različitih, ali srodnih ontologija i implementacija XSL transformacijom) i prevazilazi tematiku ovog rada

IMS LIP je specifikacija modela podataka, koja definiše koje su to informacije o studentu neophodne edukativnom sistemu radi obezbeđivanja sledećih funkcionalnosti:

- Snimanje i upravljanje istorijskih podataka vezanih za proces, ciljeve i zadatke učenja.
- Privlačenje, motivisanje i uključivanje studenta u aktivnosti učenja.
- Otkrivanje novih edukativnih mogućnosti, relevantnih iz ugla studentskih potreba, interesovanja i ciljeva.

IMS LIP model studenta (Ilustracija 9) se može posmatrati kao stablasta struktura, kao objektni model i kao tabelarna reprezentacija. Za razliku od *PAPI Learner* modela, u svojoj strukturi uključuje ne samo podatke već i opis samog modela (meta podatke). Model definiše polja podataka i tipove koje moraju zadovoljiti podaci koji se smeštaju u ta polja. Metapodaci su (na slici su originalno naslovljeni kao *Tip sadržaja*) uključeni i obavezni (znak ?) iz nekoliko razloga: smeštanje informacija vezanih za vreme (oznaka T), identifikaciju i indeksiranje (oznaka R), privatnost i zaštitu podataka studenta (oznaka P). Lako uočljiv kompozitni pristup ukazuje da ne moraju sve informacije biti uključene u model.



Ilustracija 8: IMS LIP model studenta

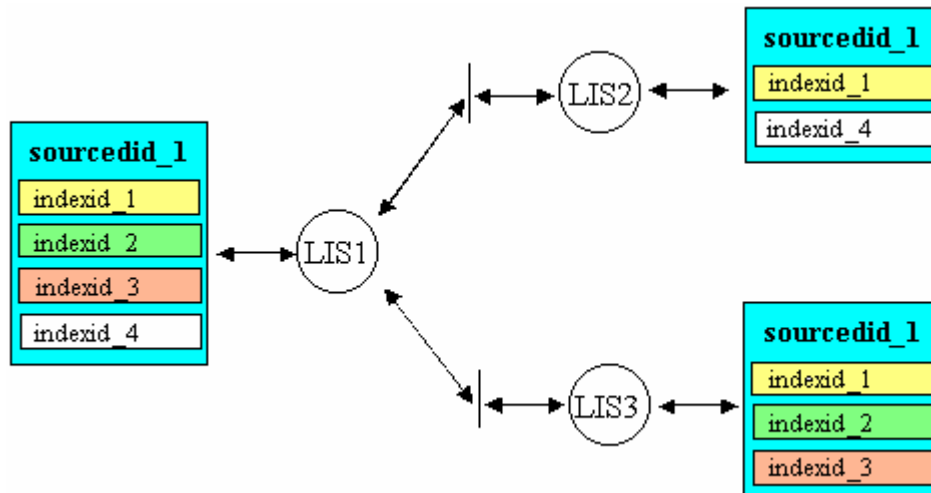
Na osnovu prethodne ilustracije može se zaključiti da IMS LIP specifikacija grupiše podatke o studentu u jedanest osnovnih informacionih tipova:

- *Identifikacioni podaci* – sadrži ključne biografske i demografske podatke o određenom studentu ili organizaciji (npr. ime, adresa, kontakt informacije).
- *Ciljevi* – sadrže opise ciljeva konkretnog studenta i to prvenstveno u domenu obrazovanja i karijere. Ovi opisi mogu sadržati i informacije relevantne za praćenje napredovanja studenta u ostvarivanju postavljenih ciljeva.
- *QCL paket* – sadrži informacije o kvalifikacijama, sertifikatima i licencama (*Qualifications, Certifications And Licenses*) koje student poseduje, a koje su izdate od strane akreditovanih organizacija (označene kao *Akreditori* u modelu

sistema). Ova kategorija sadrži formalno priznate produkte svih edukativnih i radnih aktivnosti pojedinca.

- *Aktivnosti* – sadrži zapise o svim aktivnostima studenta koje se tiču obrazovanja, bez obzira na njihov stepen završenosti. Na primer, ova kategorija beleži informacije o kursevima koje je neki pojedinac pohađao (ili trenutno pohađa), službama koje je obavljao (npr. u vojsci, zdravstvu, dobrovoljnim, humanitarnim organizacijama itd.) i rezultatima koje je ostvario na procenama znanja nakon svakog od tih kurseva.
- *Interesovanja* – informacije hobijima i drugim aktivnostima prema kojima studenta ima naročite sklonosti. To znači da student za interesovanja može da ima formalna priznanja (opisana u QCL paketu)
- *Kompetencije* – sadrži informacije koje se tiču veština, znanja i sposobnosti studenta koje je on stekao u kognitivnom, afektivnom i/ili psiho-motornom domenu. Veštine mogu biti povezane sa formalnim i neformalnim treninzima i radnom istorijom (opisanim u *aktivnostima*) i formalnim ostvarenjima (opisanim u QCL paketu).
- *Pristupačnost* –obuhvata informacije o preferencama (kognitivnim, fizičkim i tehnološkim) učenika, kao i o eventualnom postojanju nekih fizičkih/psihičkih nedostataka koje nameću potrebu za specifičnim pristupom učenju. Ove informacije u suštini opisuju sposobnost učenika da komunicira sa edukativnim okruženjem.
- *Transkript (prepis)* - ima za cilj da obezbedi sumarni pregled akademskih ostvarenja učenika. Obzirom da transkripti nemaju unificiranu strukturu i formu, specifikacija ne daje nikakve preporuke o tome kako predstaviti informacije iz ovog paketa.
- *Afilatorni podaci* – paket informacija o pripadnosti studenta različitim profesionalnim, edukativnim i drugim organizacijama (institucijama). Ovi podaci mogu biti i detaljniji (npr. pripadnost studentskoj grupi, generaciji, odeljenju, odseku i sl.).
- *Sigurnosni ključevi* – lozinke koje su dodeljene studentu za potrebe pristupa sistemu i radi obavljanja transakcija studentskih podataka između delova istog sistema i sa udaljenim podsistemima (radi ostvarivanja bezbednosti na mreži).
- *Relacije (veze)* – u ovom modelu se kao poseban informacioni paket izdvaja opis veza između pojedinih tipova informacija o studentu. Pošto su osnovna načela u dizajnu ovog modela distribuiranost i mogućnost referenciranja spolja, i pošto informacioni tipovi (paketi) nemaju vlastite identifikatore, koji bi omogućili direktno međusobno referenciranje na globalnom nivou, rešenje kreatora ovog standarda je izdvajanje opisa veza kao posebnog informacionog paketa (Ilustracija 9). Modelom su podržana dva sistema za referenciranje: *sourcedid* i *indexid*. Prvi od njih naziva se još i *identifikator zapisa studentskih podataka*. *Sourcedid* predstavlja slog labele izvora i identifikatora zapisa koji sadrži informaciju o studentu. Zamisao u IMS-u je da će jednog dana postojati jedinstveni globalni

identifikator za svaku osobu (GUID), i da će isti biti baza za formiranje *sourcedid*-a. Jedanestodelnoj strukturi *IMS LIP* modela mogu biti dodeljeni posebni brojevi kao indeksi za svaki paket posebno, unutar zapisa informacija o studentu. Ovaj koncept je imenovan kao *indexid*.

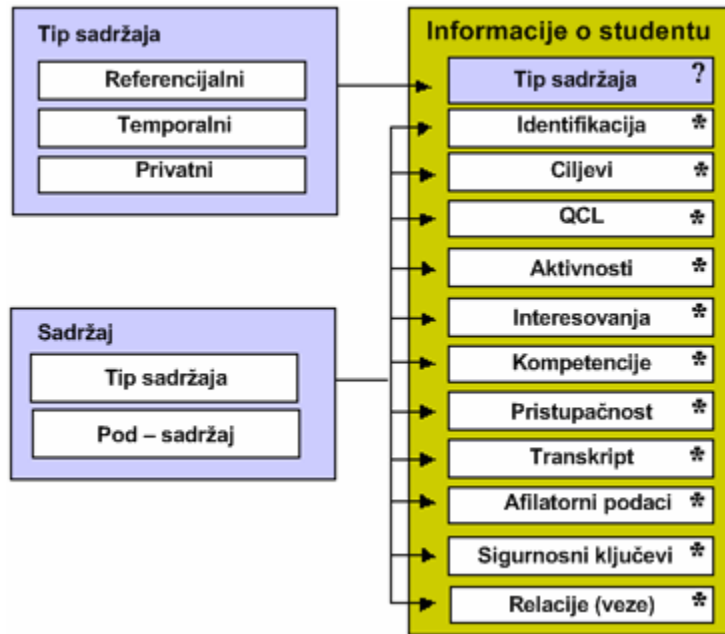


Ilustracija 9: IMS LIP – rešavanje distribuiranosti podataka studentskog modela

U distribuiranom okruženju može postojati više servera (Ilustracija 10) koji redundantno pohranjuju podatke istog studenta. Ovakav slučaj predstavljen je na prethodnoj slici. Tri servera (*LIS 1-3*) sadrže strukturane podatke studenta podeljene u četiri strukture (*indexid_1* - *indexid_4*). Prvi server (*LIS1*) sadrži kompletne podatke studenta, dok preostala dva samo delimične (*LIS2* – *indexid_1* i *indexid_4*, *LIS3* – *indexid_1*, *indexid_2* i *indexid_3*). Pošto je oznaka izvora *sourcedid_1* (koja odgovara *LIS1*), prepoznatljivo je da je poreklo podataka *LIS1*, a inicijalni slučaj korišćenja³ (u daljem tekstu SK) je da je *LIS1* distribuirao po zahtevu informacione pakete preostalim serverima. Sistem na ovaj način može da održava referencijalni integritet nad distribuiranim podacima studenta (na svi serverima se održava isti *sourcedid*), a pri daljem korišćenju sistema (razmeni različitih informacija o studentu između servera) vrši se jednostavno referenciranje preko *indexid*-ova. Ovakvim konceptom promovisana je privatnost podataka, jer na primer, serveri *LIS2* i *LIS3* mogu samo da razmenjuju informacione pakete *indexid_1* kao jedini zajednički podatak za sva tri *sourcedid_1* zapisa.

Jedna od neobičnih karakteristika *IMS LIP* modela je rekurzivnost – svaki od informacionih tipova (paketa) predstavljen je sadržajem koji dalje može da referencira na pod-sadržaj (Ilustracija 11).

³ Slučaj korišćenja (skr.SK) – u ovom materijalu se koristi kao standardizovani alat (vrsta dijagrama) za opis osnovnih funkcionalnosti sistema u UML notaciji



Ilustracija 10: Princip struktuiranja podataka u modelu

Na ovaj način moguće je ugrađivanje ekstenzija u postojeće informacione tipove bez narušavanja njihove originalne specifikacije. Ovo je značajna razlika u dizajnu *IMS LIP* u odnosu na *PAPI Learner* model radi postizanja fleksibilnosti. Rekurzivnost omogućava da, pored striktnih ograničenja nametnutih u specifikaciji informacionih tipova, dizajneri mogu slobodno proširiti model u cilju prilagođavanja istog konkretnim sistemskim zahtevima, da pri tome ne naruše njegovu rigidnu strukturu. Fleksibilnost modela se ogleda i u mogućnosti da za svaki od informacionih tipova definišemo više primeraka podataka. Na primer, za QCL paket za jednog studenta može biti više instanci – za svaki sertifikat, ili priznanje (potvrdu) poseban zapis.

IMS LIP studentski model sadrži mnogo detaljniju specifikaciju podataka studenta. U model su ugrađeni koncepti koji omogućavaju praćenje aktivnosti studenata na različitim nivoima, ciljevi edukacije, a sigurnosni ključevi i koncept veza tipova omogućavaju implementaciju privatnosti i zaštite podataka od neovlašćenog pristupa istim. *IMS LIP* standard je dizajniran radi ostvarivanja konkretnih potreba kompleksnih i distribuiranih edukativnih sistema, različitih u dizajnu, u korišćenim tehnologijama skladištenja i obrade podataka, u funkcionalnostima koje omogućavaju, da razmenjuju podatke o konkretnim studentima na siguran način uz očuvanje integriteta istih.

Robusnost modela ima veliku cenu u fazi implementacije i korišćenja sistema. Način na koji se održava integritet podataka u *IMS LIP* modelu je vrlo kompleksan, a njegova implementacija u praksi vrlo problematična. Globalni identifikatori korisnika (*GUID*) se još ne primenjuju u praksi, tako da ova karakteristika predstavlja potencijalno veliko ograničenje u pogledu ostvarivanja interoperabilnosti na globalnom nivou. Rekurzivnost predstavlja drugi potencijalni problem ovog modela. Uvođenjem ovog svojstva, odgovornost u implementaciji proširenja prebačena je na stranu korisnika modela. Ako potrebno da se struktura osnovnog modela dublje strukturi, uvode se nove ekstenzije koje povećavaju broj rekurzivnih funkcionalnih poziva. Iako je primena rekurzivnosti osobina dobrog dizajna, njena pogrešna upotreba u fazi eksploatacije modela stvaraće

probleme i može dovesti do odustajanja od korišćenja modela. Obzirom da je model projektovan da se koristi u serverskim aplikacijama, pretraživački algoritmi na velikom broju zapisa mogu dovesti do osetnog pada performansi sistema.

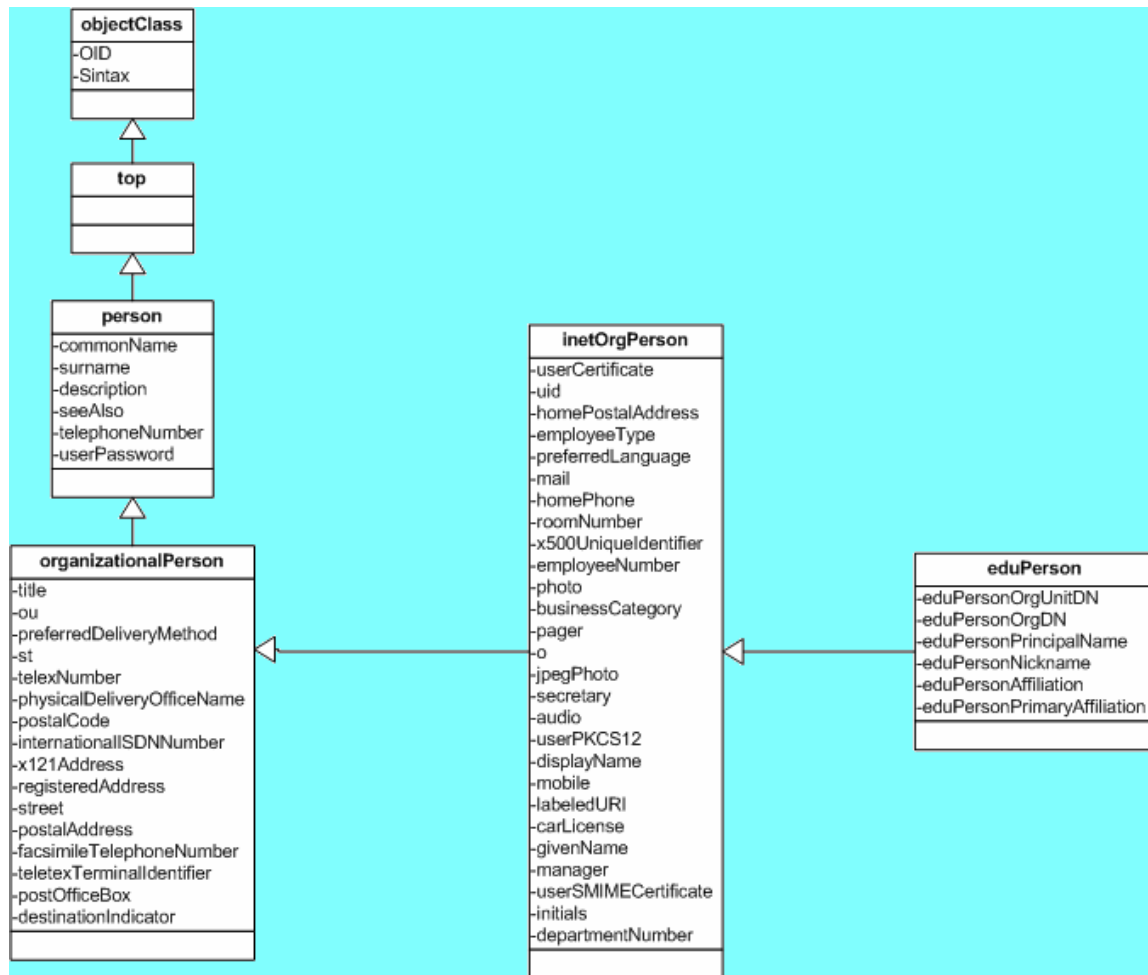
3.1.3 EduPerson

EduPerson studentski model je projektovan za potrebe visokoškolskog obrazovanja u SAD. U momentu prve specifikacije (februar 2001.godine), udruženje istraživačkih institucija *Internet2* [25] i *EDUCAUSE* [26] su isticale da je glavni motiv ove inicijative stvaranje jedinstvenog obrasca za izgradnju institucionalnih direktorijuma (za kontekst ovog rada to je pohranjivanje podataka studenata). Pošto sve edukativne institucije u SAD uglavnom razvijale sopstvene, međusobno nekompatibilne modele, a imale su sve veću potrebu razmene informacija o studentima, *EduPerson* je predstavljen kao specifikacija liste atributa koju bi različite visokoškolske institucije trebalo da podrže u studentskim modelima svojih informacionih sistema. Nekoliko univerziteta u SAD-u, među kojima se nalaze Univerzitet u Viskonsinu (engl. University of Wisconsin) i poznati institut za tehnologiju u Masačusetsu (engl. Massachusetts Institute of Technology – MIT), učestvuju u relizaciji ovog zahtevnog projekta (poslednja verzija standarda je objavljena aprila 2006.godine).

Prema *EduPerson* specifikaciji, skladišta podataka, bez obzira da li se radi o tabelarnom pohranjivanju u sistemima za upravljanje bazama podataka, ili o datotečnom čuvanju podataka u fajl sistemima, nazivaju se *direktorijumi*. *EduPerson* atributi studenata su dizajnirani da omoguće napredno pretraživanje direktorijuma. Pomenuta specifikacija sadrži skupove elemenata podataka, ili svojstava, o pojedincima unutar visokog obrazovanja, zajedno sa preporukama o sintaksi i značenju podataka koji mogu biti pridruženi pomenutim svojstvima [27]. Informacije koje su sadržane u direktorijumu su organizovane kao objekti klasa i atributi. Svaki imenovani atribut sadrži specifični element podataka (npr. adresa, afilacioni podaci i sl.). Jedan objekat klase je definisan kao skup povezanih atributa i predstavlja poseban tip podataka koji je sastavni deo direktorijuma.

EduPerson model se razvija u okviru šireg društvenog konteksta, i ima izraženu hijerarhijsku strukturu (Ilustracija 12). Dekomponovan je na šest klasa u lancu nasleđivanja. Osnovna klasa je generički objekat (*objectClass*), kojim se definiše jedinstveni objektni identifikator u sistemu i sintaksa koja se koristi radi manipulacije podacima. Ovakvim pristupom omogućeno je da se podaci studenta standardizovano serijalizuju, prenose mrežom, a pristupa im se podjednako preko API-ja i preko Web servisa. Sledeća klasa u lancu nasleđivanja je imenovana kao *top* klasa i ne sadrži podatke, ali je bitna radi održavanja koncepta direktorijuma. Ona podržava koncept korena direktorijuma, što je bitno za implementaciju algoritama za pretraživanje podataka. Klasa *person* čuva osnovne identifikacione podatke. Jedan od atributa je *seeAlso* koji omogućava referenciranje na drugi direktorijum preko drugog imena (naziva naloga, korisničkog imena) osobe. Pošto je model predviđen za institucionalizovanu edukaciju, za potrebe čuvanja podataka osoba u kontekstu organizacije, dizajnirana je klasa *organizationalPerson*. Sledeća klasa u hijerarhiji je *inetOrgPerson* u čijem dizajnu je dat akcenat na attribute koji su važni za komunikaciju posredstvom mreže (internet, ili

intranet). Iz ove klase je direktno izvedena klasa koja sadrži attribute vezane za obrazovanje – *eduPerson*.



Ilustracija 11: EduPerson studentski model

Pored nasleđenih atributa *eduPerson* sadrži šest atributa od kojih je osnovni *eduPersonOrgDN* (naziv organizacije kojoj osoba pripada). Svi atributi imaju isti prefiks – *eduPerson*. Sa atributima *cn* i *sn* (na slici *commonName* i *surname*) iz klase *organizationalPerson*, atribut *eduPersonOrgDN* formira osnovu za korišćenje klase *eduPerson* u aplikacijama. Atributi *eduPersonAffiliation* i *eduPersonPrimaryAffiliation* ukazuju na uloge koja osoba ima u edukativnoj instituciji (na primer student, saradnik, nastavnik i sl.) i oni se moraju razlikovati od atributa *title* nasleđenog iz klase *organizationalPerson*.

EduPerson specifikacija struktuirala attribute na sledeće kategorije:

- *Postojeći atributi* – standardni komercijalni atributi, nasleđeni od objekata klase *person*, *organizationalPerson* (definisani standardom X.521) i klase *inetOrgPerson* (definisani standardom RFC 2798); (npr. imena, e-mail adrese, bezbednosna podešavanja koji se nalaze u klasi *inetOrgPerson*).

- *Novi atributi* – to su zapravo svi atributi klase *eduPerson*. Namena ovih atributa je konkretna - unapređivanje saradnje (razmene informacija) između edukativnih institucija. Za svaki od attribute *EduPerson* specifikacija detaljno definiše sintaksu, značenje i uputstvo za korišćenje.

Kao kod *IMS LIP*-a, projektanti *EduPerson* modela su odlučili da reše kompleksan problem identifikatora. Za tu svrhu iskorišćen je koncept hijerarhijske strukture direktorijuma koji je saglasan sa specifikacijom RFC 3061 [29] (Ilustracija 13).

```
urn:oid:1.3.6.1
urn:oid:1.3.6.1.4.1
urn:oid:1.3.6.1.2.1.27
URN:OID:0.9.2342.19200300.100.4
```

Ilustracija 12: Primeri identifikatora (RFC 3061) koja se koristi u *EduPerson* modelu

Identifikatori su zapravo kompoziti naziva (zapravo ID-brojeva, koji se mogu mapirati u nazive i imena radi lakše manipulacije od strane ljudi – korisnika) direktorijuma i ID-brojeva konkretnih podataka (literalni – u ovom slučaju to su rastuće celobrojne vrednosti, ili randomizovani nizovi karaktera).

Jedan identifikator je ujedno i jedan element informacije, koji je specijalno dizajniran da razlikuje svaki ulazni podatak od onih koji su mi slični i da ga skladišti u poseban skup. Zajedničko ulaznim podacima je i to da sadrže nekoliko različitih identifikatora, koji se koriste u različite svrhe ili su generisani od strane različitih izvora informacija. Identifikator ima određeni broj karakteristika koje predstavljaju pomoć kod određivanja načina na koji će biti upotrebljen [28]. Osnovne osobine identifikatora su:

- *Perzistentnost* – predstavlja vremensko trajanje toku koga je identifikator pouzdano povezan sa osnovnim ulaznim podatkom. Kratkotrajni identifikatori mogu biti povezani samo u toku trajanja jedne sesije, dok su nepromenljivi identifikatori povezani sa njihovim ulazima u toku celog životnog ciklusa.
- *Privatnost* – neki identifikatori su dizajnirani tako da sačuvaju najviši stepen privatnosti i zabraniti sposobnost višestrukim nepovezanim primaocima da porede vrednosti međusobno povezanih osnovnih aktivnosti. Takvi identifikatori, kao što je REQUIRED, mogu biti nerazumljivi i da im uz to nedostaje osnovna povezanost sa ostali identifikatorima. Ovakav pristup dozvoljava deljenje identifikatora između većeg broja povezanih korisnika i to jedino u slučaju ako ih proizvođač atributa smatra ekvivalentnim jedinstvenom identifikatoru po pitanju čuvanja privatnosti.
- *Jedinstvenost* – jedinstveni identifikatori su oni koji su jedinstveni u okviru prostora imena koji govori i identitetima provajdera servisa koji je kreirao datu vrednost. Globalno jedinstveni identifikator je onaj kome je namenjeno da bude jedinstven posmatrajući sve instance atributa zastupljenih kod svih proizvođača.
- *Razduživanje* – mnogi identifikatori ne garantuju da vrednost koja im je pridružena neće biti ponovo korišćena. Ponovno korišćenje podrazumeva dodelu vrednosti jednog identifikatora nekom osnovnom ulazu, a potom dodeljivanje iste vrednosti nekom drugom ulaznom podatku u bilo kom trenutku vremena.

Neophodno je postojanje celog skupa zahteva koji diktiraju politiku koja ne dozvoljava ovakve propuste.

Na identičan način rešavaju se i vrednosti ostalih atributa studentskog modela. Sledeći primer (Fragment 1) prikazuje način serijalizacije atributa u *EduPerson* modelu. Predstavljen je fragment XML fajla koji omogućava iščitavanje podatka o korisničkom imenu nadređenog (*eduPersonPrincipalName*) posmatranog studenta (prikazano ime je *cantor.2*).

```
<saml:Attribute AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri"
  AttributeName="urn:mace:dir:attribute-def:eduPersonPrincipalName">
  <saml:AttributeValue Scope="osu.edu">cantor.2</saml:AttributeValue>
</saml:Attribute>
```

Fragment 1: Primer imenovanja u *EduPerson* modelu

Direktorijumski pristup je predstavljen vrednošću *Scope*, što znači da je ovo korisničko ime (*cantor.2*) jednoznačno definisano na mrežnom domenu *osu.edu*, čime je ograničen doseg imena. Vrednost *AttributeNamespace* definiše kontekst u kome treba da se tumači predstavljeni podatak (*cantor.2*).

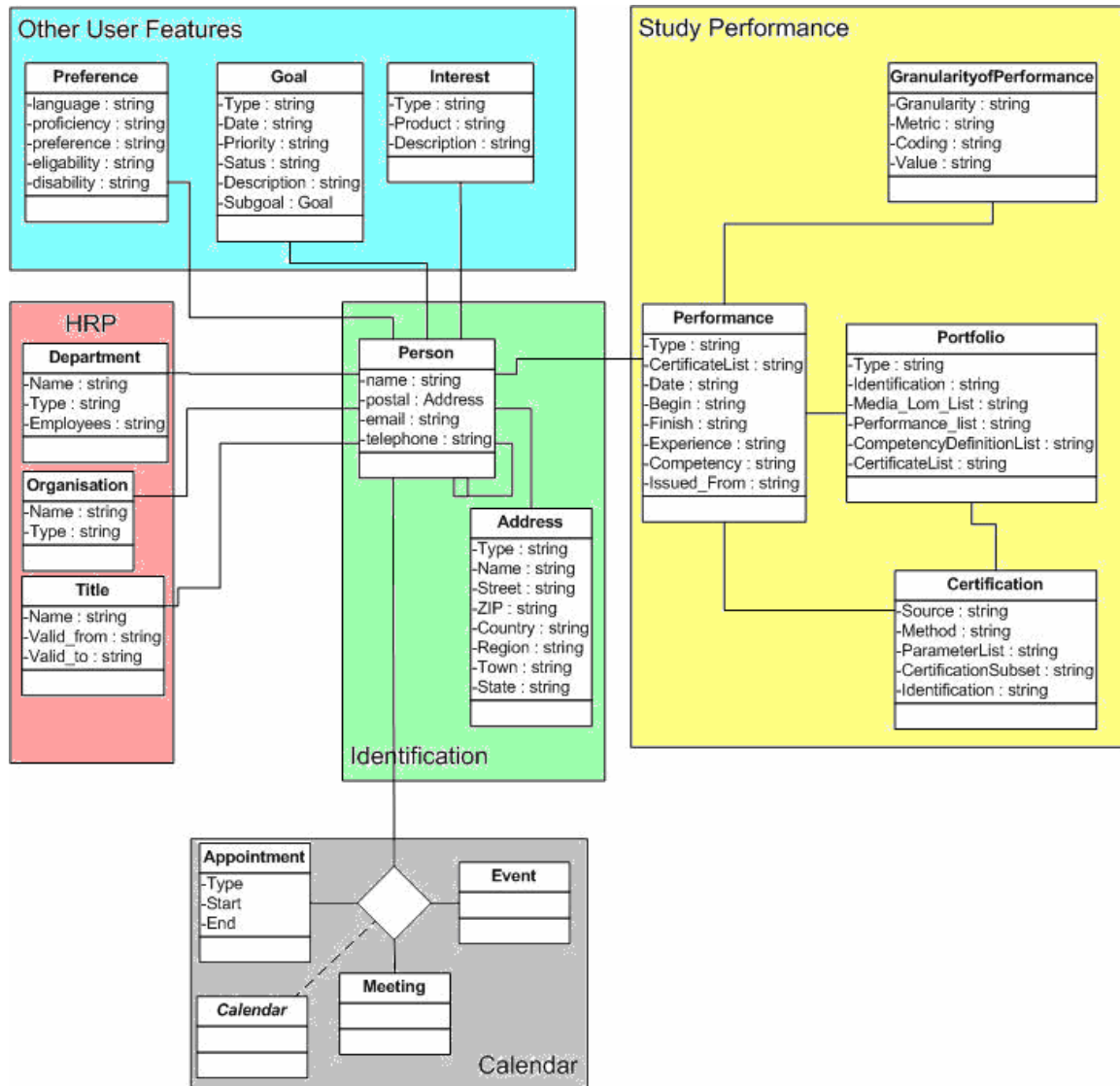
Za razliku od prethodnih modela, *EduPerson* studentski model je bolje hijerarhijski struktuiran uz izbegavanje rekurzivnosti [28]. Značaj ovog modela je nastojanje da se postigne potpuna standardizovanost, i time potpuna interoperabilnost u pogledu razmene konkretnih podataka studenata između institucija u društvu. Druga značajna osobina ovog modela je da su podaci vezani za obrazovanje hijerarhijski odvojeni od opštih podataka studenta, važnih za njegovu egzistenciju u široj društvenoj zajednici (npr. na radnom mestu, na globalnoj mreži, neobrazovnim institucijama i sl.). Hijerarhijska organizovanost pored olakšavanja administriranja podataka (koncept identifikatora), pojednostavila je i rešavanje zaštite privatnosti podataka studenata. Hijerarhijski nivoi mogu da se razmatraju kao i nivoi zaštite. Na primer, osnovni podaci osobe se čuvaju na nivou klase *person*, dok podaci važni za poslovnu komunikaciju su na nivou klase *organizationalPerson*. U tom smislu, ovaj model je razvijan korišćenjem SAML-a (*Security Assertion Markup Language*) [30].

Na žalost, uprkos velikom broju atributa, postoji samo šest koji su definisani radi razmene u obrazovnom kontekstu – atributi klase *eduPerson*. Respektujući prethodne modele (*PAPI Learner* i *IMS LIP*) Ovi atributi nisu dovoljni da obuhvate sve podatke studenta u procesu učenja. Prema dokumentaciji, proširljivost je moguća, ali je osnovni model sam po sebi već kompleksan (naročito iz aspekta implementacije), tako da u praksi još uvek nema ekstenzija ove specifikacije. Preporuka kreatora je da se, umesto proširenja postojećih klasa, definišu nove i dodaju u hijerarhiju *eduPerson* modela.

3.1.4 Studentski model ELENA projekta

Iako se radi o trogodišnjem istraživačkom projektu koji je završen (septembar 2002 – januar 2005), *ELENA* projekat [31] je značajan zbog svojih dostignuća koje je rezultiralo proširivanjem specifikacija u gotovo svim standardima za modelovanje studenta. *ELENA* projekat je u potpunosti usmeren prema semantičkom Web-u, čija je pojava bila uslovljena pojavom Web servisa, i koji predstavlja jednu od glavnih pravaca razvoja

globalne mreže. Cilj *ELENA* projekta je istraživanje o dizajnu pametnog okruženja za učenje, čiji je centralni element [21] dinamički profil studenta (Ilustracija 14).



Ilustracija 13: Studentski model *ELENA* projekta

Ovaj studentski model se zasniva na dva postojeća standarda: *PAPI Learner* i *IMS LIP* studentski model (prethodno opisana u ovom poglavlju). U *ELENA* modelu profil studenta je opisan preko 5 grupa atributa:

- *Identifikacioni atributi (Identification)* – osnovni podaci o studentu, istoimena grupa kao kod *IMS LIP*-a, dok u *PAPI Learner* modelu je isti skup predstavljen grupom *kontakt* informacija.
- *Performanse na studijama (Study Performance)* – u potpunosti opisane kompetencije, ostvareni rezultati u toku školovanja, kvalifikacije; performanse mogu da se izvede zahvaljujući klasi *GranularityofPerformance*, tako da je omogućeno da sistem sam zaključuje o performansama na osnovu pojedinačnih rezultata.

- *Podaci o zaposlenju (HRP)* – ovom grupom su obuhvaćeni podaci kao što su opis organizacije, odeljenja, radnog mesta, zvanja i roka važenja licence zvanja;
- *Kalendar* – grupa koja ne postoji u prethodno opisanim modelima; slična je konceptu aktivnosti u *IMS LIP* standardu, ističe dinamički karakter modela i podržava scenario u kome personalni agent (aplikacija, Web servis) pomaže studentu da se formalno i neformalno socijalizuje, ostvaruje kontakte i sarađuje u edukativnom procesu sa drugima.
- *Ostale karakteristike korisnika* – ovom grupom su obuhvaćeni ciljevi, interesovanja (preuzeti iz *IMS LIP* specifikacije) i preference (preuzeto iz *PAPI Learner* modela);

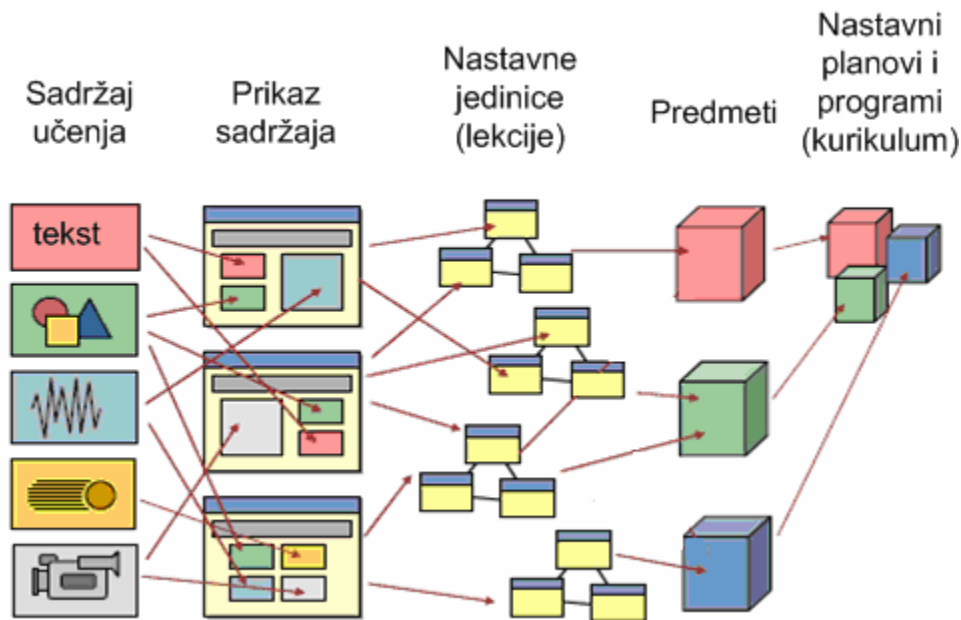
Iako su u studentskom modelu *ELENA* projekta atributi većim delom preuzeti iz drugih standardnih modela, realizovana je mnogo bolja sistematizacija podataka. Omogućeno je da se o performansama (kompetentnostima) izvode zaključci, a ne kao u drugim modelima – koriste eksplicitni transkripti (*IMS LIP*), ili portfolija (*PAPI Learner*). Model omogućava da edukativni sistem da ocenu tražene kompetentnosti zaključivanjem nad granuliranim performansama. Na taj način može da se smanji, ili zameni ljudski faktor u ovoj vrsti odlučivanja (podložnoj subjektivnom ocenjivanju). Slično *eduPerson* modelu dodati su atributi za povezivanje studenata u širi društveni kontekst – radno okruženje. Cilj ovog proširenja je podrška konceptu doživotnog učenja, koje treba da proistekne iz potrebe za profesionalnim usavršavanjem.

3.2 Modelovanje i pakovanje sadržaja učenja

Sadržaji učenja su resursi koji sadrže domensko znanje (koje se izučava), koji koriste studenti, čijim se korišćenjem odvija proces učenja, a kojima se pristupa posredstvom edukativnog sistema. Sadržaje učenja najčešće kreiraju nastavnici, saradnici, i/ili domenski eksperti, u edukativni sistem ga na odgovarajući način ugrađuju sistem administratori ili posebno obučeni operatori, a kako što je već rečeno – koriste studenti. Sadržaji učenja su najčešće predstavljeni tekstualnim dokumentima, elektronskim knjigama, ilustracijama, video i audio zapisima, materijalima za testiranje i animacijama.

Tendencija u savremenim edukativnim sistemima je da sadržaji učenja budu nezavisni u odnosu na organizovanje nastave (Ilustracija 14). Sadržaji treba da u sistemu egzistiraju potpuno samostalno, da budu opisani na što potpuniji i standardizovan način. Da bi postali višestruko upotrebljivi u nastavi, sadržaji treba da budu što bolje fragmentirani. Na primer, bolje je da postoji puno manjih, dobro opisanih (naziv, ključne reči, tip fajla, verzija, šira, ili uža naučna oblast i sl.) tekstualnih fajlova, od kojih svaki opisuje neki pojedinačni koncept, nego jedan fajl koji će sadržati tekstualni materijal čitavog predmeta. U prvom slučaju, fragmenti mogu da se uključuju u različite predmete, na različite načine, a u drugom ne. Podaci kojima se vrši opis sadržaja nazivaju se *metapodacima* (podaci o podacima) sadržaja. Vrednosti metapodataka (opisi) služe lakšem pronalaženju željenih sadržaja na repozitorijumima (skladištima resursa) od strane fizičkih korisnika, ali i aplikacija. Opisane i fragmentirane, sistem može da prikazuje sadržaje na različite načine i u različitim kontekstima. Isti resurs može da se u jednom kontekstu koristi za učenje, a u drugom za detaljnije objašnjavanje nekih drugih

pojmovi, koji su opisani drugim resursima, ili na primer, kao objašnjenje (pomoć) za načinjenu grešku studenta pri testiranju.

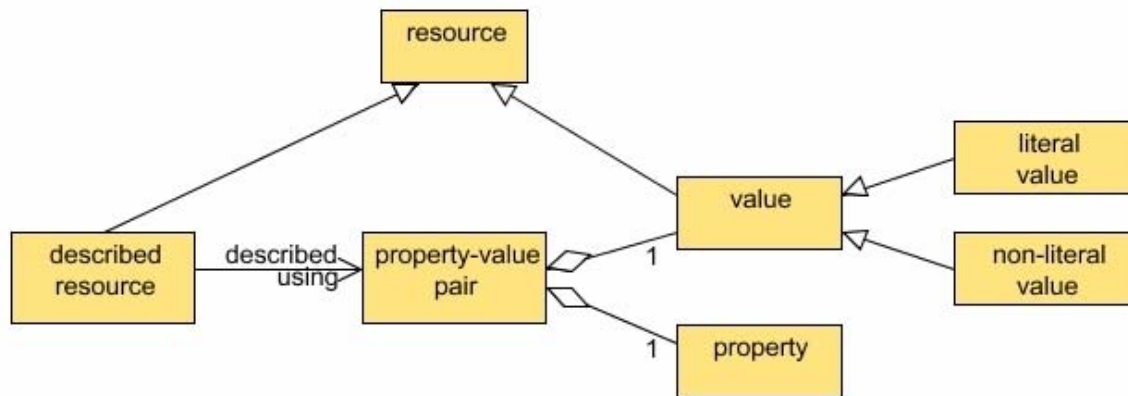


Ilustracija 14: Odnos sadržaja učenja i organizacije procesa učenja

Strukturu predmeta sačinjava mreža nastavnih jedinica (usmeren graf koji oslikava njihovu uzajamnu zavisnost). Sadržaji se referenciraju – povezuju za nastavne jedinice, tako da je njihova veza tipa *više – više* (isti sadržaj može da se koristi u različitim lekcijama, i jedna lekcija može da sadrži različite sadržaje). Ove veze su fleksibilne, mogu se uspostavljati i raskidati prema potrebi. Prikaz sadržaja predstavlja poseban koncept, koji može da zavisi od tipa sadržaja (tekst, slika, video i td.), konteksta korišćenja, potreba konkretnog studenta i prezentacionih mogućnosti sistema. Prikaz sadržaja generiše edukativni sistem na osnovu navedenih faktora.

3.2.1 Dublin Core

Dublin Core Metadata Initiative (DCMI) [34] je otvorena organizacija koja je uključena u razvoj standarda metapodataka za opis resursa različite namene, prvenstveno radi postizanja interoperabilnosti na globalnoj mreži. Njihov standard ima zvaničan naziv *DublinCore*, datira iz 1995.godine, nazvan je po mestu Dublin-u (Ohio, USA), u kome je održana prva konferencija ove inicijative, i po engleskoj reči *Core* (u kont. *suštinski*) kako bi se ukazalo da ovaj standard treba da predstavlja nezaobilaznu osnovu za opisivanje resursa. Ovo je i prvi standard korišćen za opisivanje resursa učenja, iz kog su proistekli svi ostali standardi kao njegova proširenja. Postoje dve aktuelne verzije standarda – verzija 1.0 i verzija 1.1. Standard je razvijen na generalnom *DCMI* modelu za opisivanje resursa (Ilustracija 15).



Ilustracija 15: DublinCore model resursa

Resurs je generalni koncept koji se specijalizuje opisivanjem uređenim parovima svojstvo – vrednost. Ovaj par predstavlja kompozit koji može da sadrži jedno svojstvo i njemu odgovarajuću vrednost. Vrednost može da bude literal, ili struktura za sebe. Oba navedena entiteta (svojstvo i vrednost) predstavljaju klase za sebe. U *DublinCore* specifikaciji za svojstva resursa koristi se termin *element*. Definisano je 15 elemenata:

- *Contributor* (učesnik) – može da bude osoba, organizacija, ili servis, koji ima učešća u generisanju resursa.
- *Coverage* (kontekst) – kontekst (geografski, vremenski, organizacioni ili domenski) u kome je resurs upotrebljiv.
- *Creator* (kreator) - može da bude osoba, organizacija, ili servis (entitet), odgovoran za kreiranje resursa.
- *Date* (datum) – vremenski trenutak ili period povezan sa događajem u životnom ciklusu resursa.
- *Description* (opis) - apstrakt, tabela sadržaja, grafička reprezentacija, ili tekst u slobodnoj formi koji bliže opisuje resurs.
- *Format* (tehnički format) – tip fajla resursa, fizički medijum, veličina i trajanje resursa.
- *Identifier* – jedinstveni identifikator resursa, po kojem se razlikuje od svih ostalih resursa u kontekstu sistema.
- *Language* (jezik) – jezik koji se koristi u resursu.
- *Publisher* (izdavač) - može da bude osoba, organizacija, ili servis (entitet), koji omogućava da resurs postane dostupan korisnicima.
- *Relation* (relacija) – relacija označava povezan resurs.
- *Rights* (prava) – informacije o pravima korišćenja, modifikovanja, vlasništva i distribucije resursa.
- *Source* (izvor) – resurs se povezuje relacijama sa drugim resursima. Može da bude izveden iz njihovih delova ili celina. U tom slučaju, izvori su važni za konstrukciju posmatranog resursa.

- *Subject* – tema resursa, koja može da bude opisana i ključnim rečima, frazama, ili klasifikacionim kodovima.
- *Title* – naslov resursa. Važno za pretragu i prezentaciju u kontekstu objašnjavanja.
- *Type* – tip resursa, za koji se preporučuje korišćenje rečnika tipova.

Uz primenu standardizovanih rečnika kao skupova predefinisanih vrednosti svojstava (npr. postoje zvanične preporuke za: rečnik geografskih pojmova – *TGN* [36], rečnik tipova sadržaja – *DCMITYPE* [35], *W3CDTF* datumske profile standarda *ISO 8601* [37], rečnik tipova medijuma na Internetu *MIME* [38], rečnik za opis jezika *RFC 4646* [39]), *DublinCore* specifikacijom se potpuno definišu resursi učenja. Zahvaljujući kompletnoj specifikaciji, ovaj standard je opšte prihvaćen kao platforma za druge edukativne standarde, koji predstavljaju proširenja *DublinCore* specifikacije.

Svako svojstvo (*element*) je struktura za sebe i opisana je skupom od deset atributa. Ovi atributi predstavljaju *meta-metapodatke DublinCore* standarda. Šest atributa ima zajedničke vrednosti za sve *DublinCore* elemente (Tabela 6) koji opisuju isti resurs (sadržaj).

Tabela 6: Zajednički atributi *DublinCore* elemenata

NAZIV	VREDNOST
Version:	1.1
Registration Authority:	Dublin Core Metadata Initiative
Language:	en
Obligation:	Optional
Datatype:	Character String
Maximum Occurrence:	Unlimited

To znači da u fajlu koji opisuje resurs, nije moguće da se nađu dva elementa (npr. *creator* i *publisher*), koji imaju različiti atribut *Version*. Preostala četiri atributa imaju različite vrednosti kod različitih elemenata. Ovi atributi su:

- *Name* – naziv elementa.
- *Identifier* – jedinstveni identifikator dodeljen elementu. Obično se radi o skraćenom obliku naziva elementa.
- *Definition* – iskaz koji jasno opisuje concept elementa.
- *Comment* – naznaka usmerena na primenu elementa.

Ovi atributi identifikuju (*name* i *identifier*) i definišu elemente, i daju praktične preporuke o kontekstu i načinu korišćenja elemenata. Na sledećoj tabeli (Tabela 7) dat je primer elementa *Identifier* koji ima jasnu definiciju da on treba jednoznačno da određuje resurs na globalnoj mreži. U komentarima je sugerisano da se za označavanje koriste stringovi, ili numeričke vrednosti, a kao primeri identifikacionih sistema navedeni su *ISBN*, *URI*, *URL* i *DOI*.

Tabela 7: Primer definisanja atributa *DublinCore* elementa *Identifier*

Name:	Resource Identifier
Identifier:	Identifier
Definition:	An unambiguous reference to the resource within a given context
Comment:	Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Example formal identification systems include the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN).

Da bi opis resursa bio mašinski čitljiv, pored navedenih osobina (specifikacija elemenata i atributa), potrebno je da opis bude predstavljen u nekom standardnom formatu (jeziku), koji omogućava struktuiranje podataka, njihovu serijalizaciju i transfer između slabospregnutih, ili nezavisnih aplikacija/softverskih komponenti. U najčešćem slučaju, korišćen je XML *tag*-ovan jezik. Na sledećoj ilustraciji (Fragment 2) dat je fragment opisa resursa (*index.html*) kome se mrežno pristupa preko *URI* (<http://whyfiles.org/107flood/index.html>).

```
<?xml version="1.0" encoding="UTF-8"?>
<dcxf:descriptionSet xmlns:dcxf="http://dublincore.org/xml/dc-xml-full/2007/06/19">
  <dcxf:description dcxf:resourceURI="http://whyfiles.org/107flood/index.html">
    <dcxf:statement dcxf:propertyURI="http://purl.org/dc/elements/1.1/title">
      <dcxf:literalValueString xml:lang="en-US">Flood of Evidence</dcxf:literalValueString>
    </dcxf:statement>
    ...
  </dcxf:description>
</dcxf:descriptionSet>
```

Fragment 2: Primer opisivanja jednog edukacionog resursa u *DublinCore* XML formatu

U primeru je predstavljen jedan element opisa – svojstvo *naslov* (*title*) resursa. Vrednost naslova je definisana kao string na engleskom jeziku (*Flood of Evidence*). Predstavljeni *tag*-ovi (*description*, *statement* i *literalValueString*) imenovani su u okviru skupa za opis (*descriptionSet*) koji ima prefiks (proctor imenovanja – *XML namespace*) *dcxf* (označava da je *DublinCore* opis u XML formatu) i koji je definisan na Web adresi (URL) <http://dublincore.org/xml/dc-xml-full/2007/06/19>. Plava, crvena i zelena vertikalna linija su dodate da bi hijerarhijska struktura dokumenta bila lakše uočena.

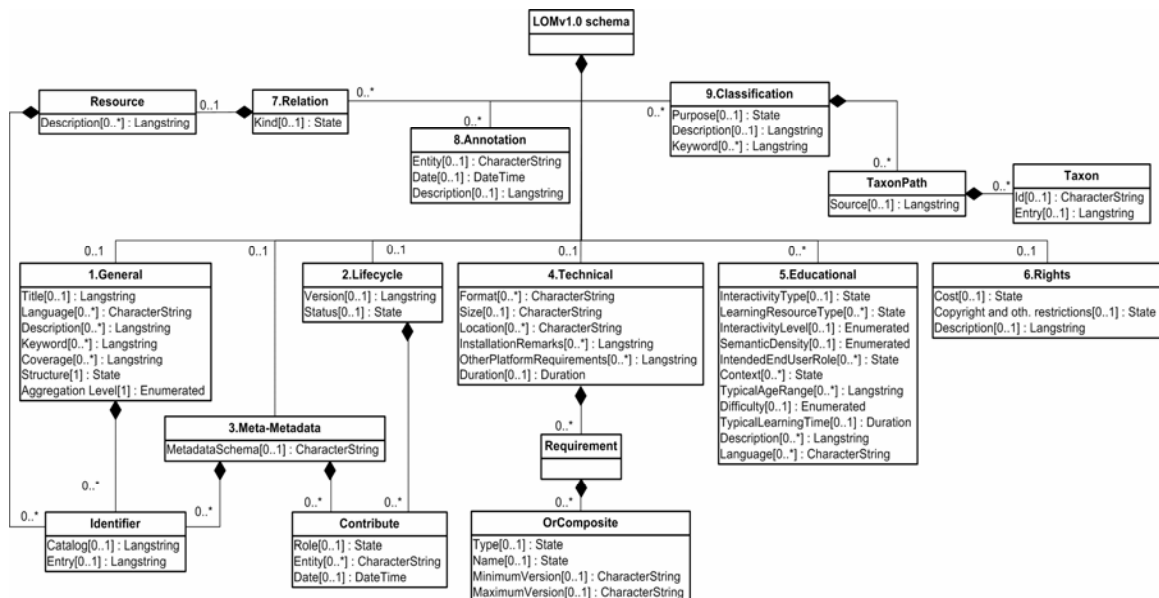
Često referenciranje *DublinCore* specifikacije (u literaturi, naučnim krugovima, IT industriji) ukazuje na njen dobar dizajn i upotrebljivost u različitim, a za istraživanje interesantnim – edukativnim kontekstima. Pored navedene specifikacije postoji *Dublin Core Metadata Element Set* (opšte poznat kao *Simple Dublin Core*, standardizovan kao *ANSI/NISO Z39.85 – 2001*). Ova specifikacija obezbeđuje jednostavniji, mnogo manji set elemenata, pogodnih za deljenje metapodataka kroz široku mrežu različitih servisa. Naspram *Simple Dublin Core* –a, *DCMI* specifikacija, opisana u ovom poglavlju, ima mnogo veću semantičku preciznost i poznata je pod nazivom *Qualified Dublin Core*.

Opšta namena ovog standarda može da predstavlja nedostatak, jer za svaku posebnu specijalizaciju treba da se prave proširenja, koja moraju striktno da podržavaju koncepte *DublinCore* specifikacije. Ovo ograničenje može da bude frustrirajuće za dizajnere (*developer-e*) nove specifikacije. Drugi nedostatak takođe proizilazi iz opštosti - to je

kompleksnost specifikacije. Kao što se može videti iz navedenog primera, cena dobrog formatiranja je da je količina neinformativnog sadržaja dokumenata mnogo veća nego sam informativni sadržaj (naziv svojstva i njegova vrednost). Za veliki broj resursa (i veliki broj korisnika), pretrage po elementima opisa mogu da utiču na pad performansi sistema.

3.2.2 LOM (Learning Object Metamodel)

IEEE LOM Standard (IEEE 1484.12.1 – 2002) je prvi akreditovani standard za opisivanje resursa (sadržaja učenja) u oblasti edukativnih tehnologija. Osnovni koncept na kome se zasniva *LOM* specifikacija je objekat učenja (u daljem tekstu *LO* – *Learning Objects*). *LO* je resurs (sadržaja učenja), koji pored vlastitog sadržaja (fajlovi različitog tipa), ima definisan opis preko standardnih svojstava i njihovih vrednosti. Ova svojstva se još nazivaju metapodacima *LO*. Prethodno opisana *Dublin Core* specifikacija takođe opisuje resurse posredstvom standardizovanih opisa, ali je to specifikacija opšte namene, tako da *LOM* predstavlja njeno proširenje elementima i atributima prilagođenim specifičnim potrebama studenata i nastavnika za relevantnim edukativnim sadržajima.



Ilustracija 16: Model LOM standarda

LOM standard ima hijerarhijsku strukturu (Ilustracija 16) mada agregacija predstavlja osnovnu vezu između koncepata modela (za razliku od tipične hijerarhijske strukture, u kojoj dominira veza nasleđivanja između koncepata modela). *LOM* standard grupiše elemente (svojstva resursa, vidi *Dublin Core* specifikaciju) za opisivanje *LO* u devet kategorija u skladu sa njihovom namenom:

- *Opšta kategorija (general)* –elementi kojima se opisuju karakteristike *LO* koje ne zavise od konkretnog konteksta primene *LO*. Na primer, ovoj kategoriji pripadaju elementi: jedinstveni identifikator, naslov, jezik, ključne reči i sl.
- *Kategorija životnog ciklusa (lifecycle)* –karakteristike *LO* relevantne sa aspekta opisa njegovog životnog ciklusa. To su elementi: verzija, status, i sl.

- *Kategorija meta-metapodataka (meta-metadata)* –elementi koji opisuju metapodatke *LO*, odnosno podatke kojima je anotiran *LO*. Elementima ove kategorije beleže se podaci o: korišćenju *schema*-i metapodataka, osobi koja je uradila anotaciju i/ili validaciju metapodataka i sl.
- *Tehnička kategorija (technical)* –elementi kojima se opisuju osnovne osobine *LO* iz tehničkog aspekta. Neki od elemenata ove kategorije su: format, trajanje, veličina, specifični zahtevi instalacije i sl.
- *Edukativna kategorija (educational)* – elementi namenjeni za opis edukativnih/pedagoških karakteristika *LO*. Oni opisuju, na primer, godinu studija (uzrast) kojoj je *LO* namenjen, težinu sadržaja *LO*, zahtevani stepen interakcije studenta i *LO*, semantičku gustinu *LO* (upotrebljivost u odnosu na veličinu *LO* i dužinu trajanja i sl).
- *Kategorija prava pristupa (rights)* –elementi namenjeni definisanju prava pristupa *LO*, kako bi se obezbedila zaštita prava intelektualne svojine.
- *Kategorija relacija sa drugim LO (relation)* - elementi za opis relacija specificiranog *LO* sa drugim *LO*. Relacije mogu biti različitog tipa i ukazivati da je tekući *LO* deo nekog drugog *LO*, ili da sadrži neki drugi *LO*, ili da je nova verzija/format nekog drugog *LO* i sl.
- *Kategorija označavanja (annotation)* –elementi za beleženje komentara o iskustvima stečenim primenom *LO* u edukativnoj praksi.
- *Kategorija klasifikacije (classification)* –metapodaci za opis pozicije *LO* u odnosu na neke postojeće klasifikacione sisteme.

Za svaki pojedinačni resurs (*LO*) *LOM* opis može da sadrži, ili ne, najviše jednu vrednosti za svaku navedenu kategoriju. Ove vrednosti mogu da budu različitog tipa: najčešće su to stringovi, ali pored toga mogu da se koriste i izvedeni tipovi:

- *LangString* – tip koji predstavlja uređeni par: string i oznaka jezika na kom je string napisan
- *Vocabulary* – kontrolisane liste termina (od strane nekog entiteta, ili standardom), predstavljene kao uređeni parovi *izvor-vrednost*, pri čemu je izvor naziv liste iz koje se koristi vrednosti
- *DateTime* i *Duration* – datumski i vremenski tip koji je formatiran da bude mašinski čitljiv (npr. *UTF*, *CET* i sl.).

Osnovni motiv *LOM* specifikacije je da se pronade najmanji zajednički sadržalac potreba edukativnih aplikacija i da se na osnovu toga kreira standard koji će podmiriti identifikovani skup potreba. Sve kategorije metapodataka su opcione, i korisnicima je ostavljena mogućnost proširenja osnovne *schema*-e u skladu sa svojim specifičnim potrebama. *LOM* je zamišljen da bude zajednička polazna osnova za označavanje edukativnih sadržaja metapodacima, koja će omogućiti razmenu i ponovno korišćenje tih sadržaja [40]. Praktični primeri ovakvog koncepta su sledeća proširenja: britanski *UK LOM Core* [41], kanadski *CanCore* [42], australijski *Vetadata* [43], norveški *NORLOM* [44].

```

<?xml version="1.0" encoding="UTF-8" ?>
<lom xmlns="http://ltsc.ieee.org/xsd/LOMv1p0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ltsc.ieee.org/xsd/LOMv1p0 http://www.rdn.ac.uk/oai/lom/lom.xsd">
  <general>
    <title>
      <string>developerWorks : XML</string>
    </title>
    <description>
      <string>The XML zone on the developerWorks Web site is designed for developers. You'll find tools, samples, standards
        information, education, news and events, and links to XML community forums and Web sites.</string>
    </description>
    <!-- Many other keywords snipped -->
    <keyword>
      <string>xml resources</string>
    </keyword>
    <keyword>
      <string>xml programming</string>
    </keyword>
  </general>
  <lifecycle />
  <technical>
    <format>text/html</format>
    <location>http://www-106.ibm.com/developerworks/xml/</location>
  </technical>
  <educational>
    <learningResourceType>
      <source>DCMIType</source>
      <value>Text</value>
    </learningResourceType>
  </educational>
</lom>

```

Fragment 3: Primer korišćenja LOM standarda u opisu edukativnog resursa

U praksi, *LOM* opis edukativnih resursa je najčešće predstavljen XML dokumentom (Fragment 3). XML format u potpunosti podržava hijerarhijsku strukturu opisa. Koreni *tag* (čvor) ima oznaku standarda (*lom*), i iskorišćen je za referenciranje na *LOM* specifikaciju (*namespace*). Kao što se može videti iz druge i treće linije primera, specifikacija je definisana korišćenjem XML schema-e (*xsd* format). Ovaj deo dokumenta omogućava da ostatak opisa bude mašinski čitljiv, odnosno da aplikacije mogu koristiti ove metapodatke u kontekstima pretraživanja i korišćenja resursa.

U primeru su korišćene tri kategorije opisa: generalna, tehnička i edukaciona (*tag*-ovi *general*, *technical* i *educational*). Pored toga, navedena je i kategorija životnog ciklusa (*lifecycle*), ali ne sadrži nikakve podatke (prazan *tag*). U generalnoj kategoriji predstavljeni su podaci: naziv, opis resursa i ključne reči (*tag*-ovi *title*, *description* i *keyword*). U tehničkoj kategoriji predstavljeni su podaci o formatu i lokaciji resursa, a u edukativnoj kategoriji podatak iz standardnog DCMI rečnika tipova resursa (preuzet iz *Dublin Core* specifikacije) – uređen par *izvor* – *vrednost*.

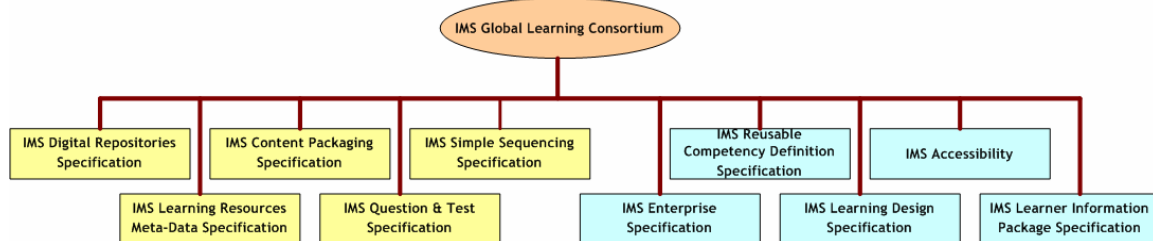
Zahvaljujući prihvatanju LOM standarda od strane formalnih institucija i organizacija koje se bave edukacijom, on predstavlja glavni međunarodni standard za opis sadržaja učenja, koji se koristi u velikom broju aktuelnih sistema za e-učenje.

3.2.3 IMS standardi za sadržaje učenja

IMS (Instructional Management System) projekat započet je 1997.godine u Velikoj Britaniji, s ciljem razvoja globalnog distribuiranog učenja na nivou visokoškolskog obrazovanja, kroz kolaborativne i kooperativne napore univerziteta, neprofitnih organizacija, agencija vlade i drugih udruženja i organizacija. U januaru 2000.godine osniva se samostalna neprofitna organizacija – *IMS Global Learning Consortium* [22], jedna od najnaprednijih u oblasti razvoja specifikacija namenjenih ostvarivanju interoperabilnosti edukativnih tehnologija. Ova organizacija iznedrila je deset, danas

intenzivno korišćene standarde u oblasti elektronskog učenja (Ilustracija 17). Polovina (pet) standarda su vezani za sadržaje (resurse) učenja (na slici u žutim pravougaonicima):

- *IMS Learning Resources Meta-Data Specification* – namenjena za opis resursa učenja u cilju pretrage i ispitivanja sadržaja.
- *IMS Content Packaging Specification* – namenjena za kreiranje i deljenje objekata učenja koji su višestruko upotrebljivi.
- *IMS Simple Sequencing Specification* – namenjena za specifikaciju raspoređivanja i prikazivanja studentima objekata učenja.
- *IMS Question & Test Specification* – namenjena da obezbedi deljivost resursa testiranja i proveravanja.
- *IMS Digital Repositories Specification* – namenjena za povezivanje sistema za učenje sa relevantnim informacionim resursima.



Ilustracija 17: Standardi *IMS Global Learning Consortium-a*

3.2.3.1 IMS Learning Resources Meta-Data Specification

IMS Learning Resource Metadata (IMS LRM, 2000) je specifikacija koju je definisala ova grupa sa ciljem da unapredi *IEEE LOM*. *IMS LRM* specifikacija se u potpunosti zasniva na elementima *IEEE LOM* standarda (time i na *Dublin Core-u*), ali ga još bolje standardizuje u smislu korišćenja različitih rečnika podataka (Tabela 8).

Tabela 8: Primer specifikacije metapodataka resursa učenja u *IMS LRM*

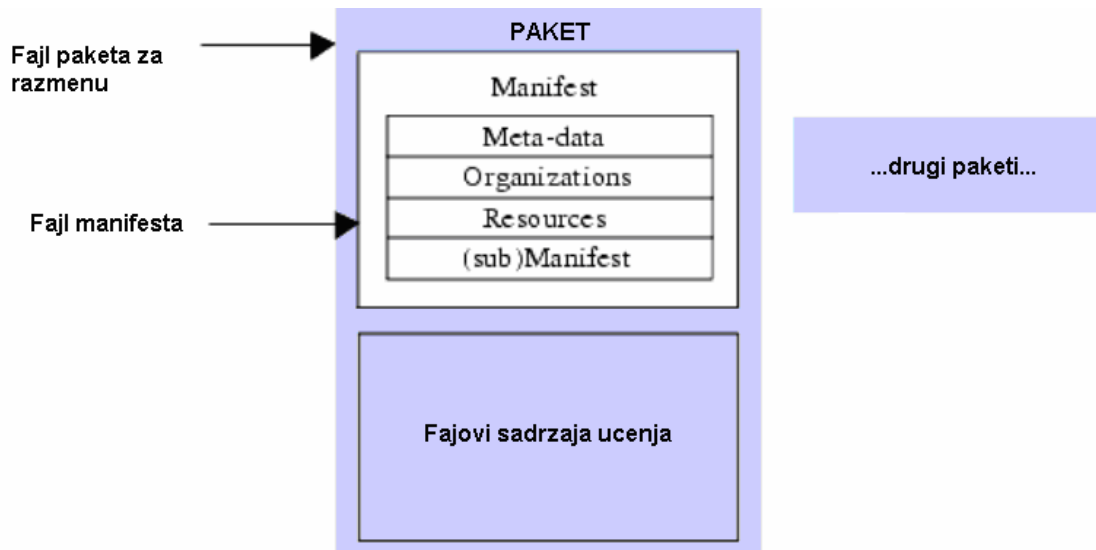
Element	Taksonomija Rečnik	Polja upotrebe	Karakteristike	Izvor
technical.format	RFC 1521	Australija više obrazovanje, Vojska SAD Obuka	Stabilan i zreo; Nacionalni/internacionalni doseg; kontrolisani rečnik	http://www.isi.edu/in-notes/fiana/assignments/media-types/media-types
	GEM format controlled vocabulary	SAD K12 (visoko obrazovanje)	Relativno stabilan i zreo; domaći-korišćenje u rastu (podskup RFC 1521 medija tipova); kontrolisani rečnik	http://www.geminfo.org/Workbench/Metadata/Vocab_Format.html
	Merlot format list		Novi rečnik, zreo; domaći-korišćenje u rastu; ograničen rečnik	http://merlot.org/search/AdvArtifactSearch.po

U *LOM* specifikaciji, za izbor vrednosti jednog svojstva resursa, date su preporuke za samo jedan standardni rečnik. U *IMS LRM* specifikaciji izvršena je sveobuhvatna analiza svih taksonomija i rečnika podataka, dostupnih preko globalne mreže. Korisnici ove specifikacije mogu da izaberu jednu od ponuđenih klasifikacija za zadato svojstvo resursa koji opisuju. Na prethodnoj ilustraciji prikazana je specifikacija za tehničko svojstvo *format*, u kojoj su ponuđena tri rečnika sa opisom: njihovi nazivi, gde se koriste,

najvažnije karakteristike i adresa (URL) na kojoj se mogu dobiti detaljni podaci o rečniku.

3.2.3.2 IMS Content Packaging

IMS Content Packaging specifikacija namenjena je postizanju interoperabilnosti za razmenu sadržaja učenja između aplikacija za kreiranje nastavnih materijala, sistema za upravljanje učenjem i okruženja za korišćenje materijala.

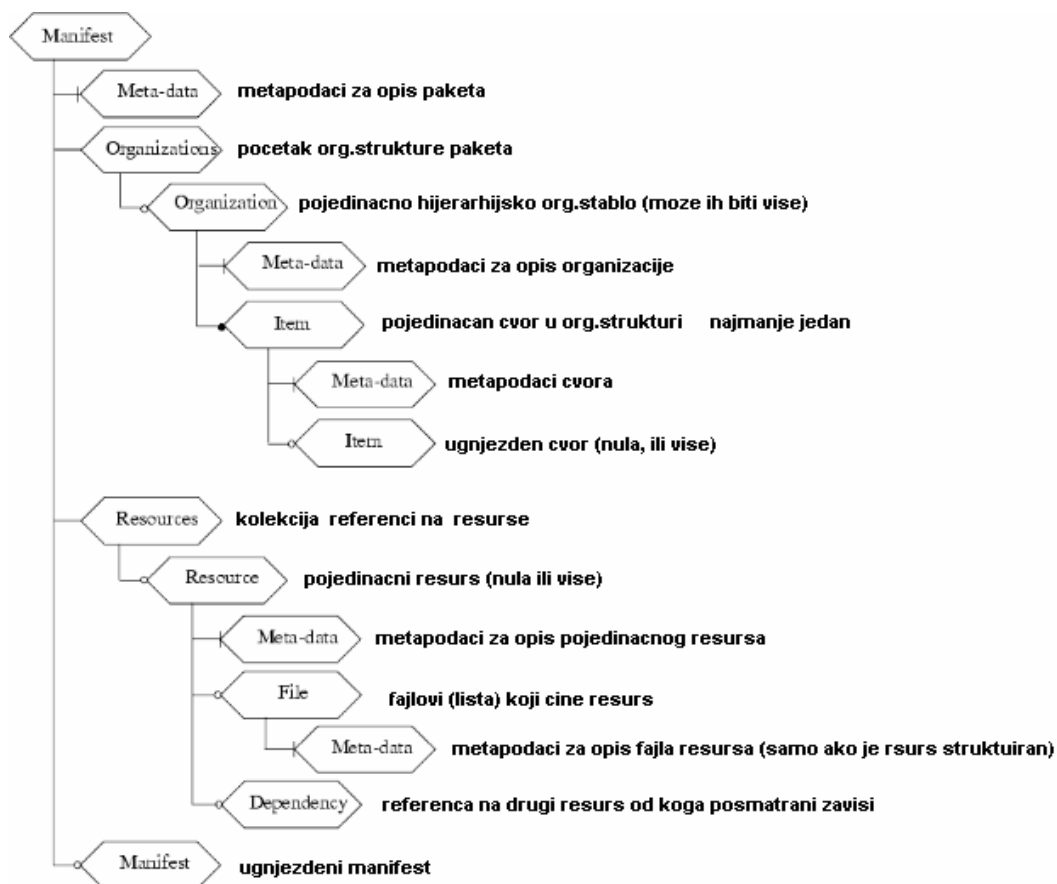


Ilustracija 18: Način pakovanja sadržaja IMS CP specifikacijom

Paket za pakovanje sadržaja je, u fizičkom smislu, arhivski fajl koji ima specificiranu strukturu. Ovu strukturu čine dva osnovna elementa (Ilustracija 18):

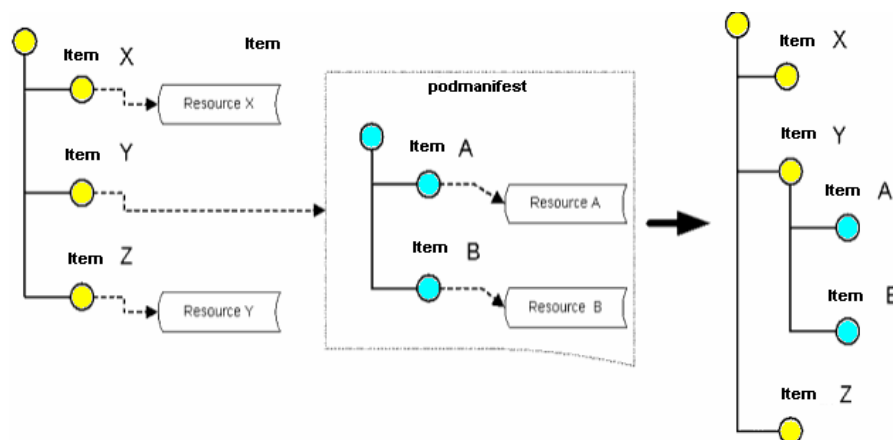
- *Manifest fajl* – predstavlja opis resursa učenja koji se nalaze u paketu za razmenu. Opis je napisan u XML formatu. Naziv fajla je *imsmanifest.xml*
- *Resursi u paketu* – to su fizički fajlovi tekstualnog, grafičkog, multimedijalnog, i drugih tipova, koji mogu biti organizovani u pod-foldere, i koji su opisani u manifestu.

Svaki manifest (*imsmanifest.xml*) sadrži opis organizacione strukture paketa i reference resursa koji su uključeni u paket (Ilustracija 19). Koreni čvor *Manifest* sadrži 2 podatka: identifikator manifesta i verziju. Metapodaci za opis paketa sadrže format opisa (korišćena *schema*), njegovu verziju i same opis. Čvor *Organizations* opisuje jednu, ili više struktura, ili organizacija u paketu (čvorovi tipa *Organization*, koji su opcioni). Čvor *Organization* mora da sadrži barem jedan čvor tipa *Item*. Namena čvora *Item* je da povezuje resurs sa organizacijom. Jednom *Item*-u odgovara tačno jedan resurs. Veza se ostvaruje preko atributa *Item*-a, koji je imenovan kao *identifierRef*. Ovaj podatak je referenca na identifikator resursa u sekciji (čvoru) *Resources*. Ako je *Item* složen, on se dalje strukturiira u pod-čvorove koji su takođe tipa *Item*.



Ilustracija 19: Struktura IMS CP paketa

U sekciji (čvoru) *Resources* prikazuje se lista resursa (čvorovi tipa *Resource*). Čvorovi za pojedinačne resurse dalje može da sadrži čvor metapodataka resursa, listu fajlova koji sačinjavaju resurs i listu zavisnosti posmatranog resursa od drugih resursa. Pored toga, manifest može da sadrži pod-manifeste (*sub-manifests*). Ovaj tip ugnježdavanja (Ilustracija 20) omogućava da se, u slučaju složenih struktura organizacije resursa i nastavnih sadržaja, izvrši dekonponovanje opisa na manje granule, koje omogućavaju agregiranje sadržaja prema potrebi.

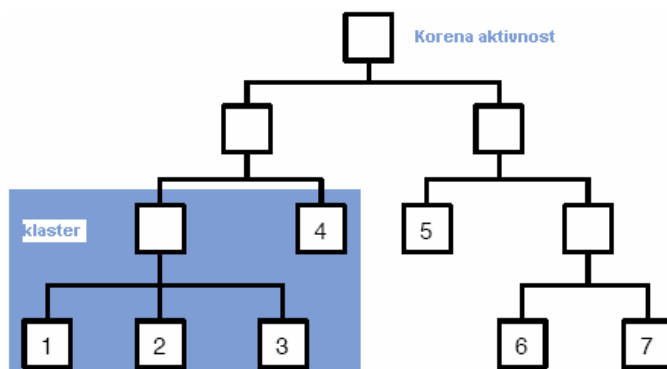


Ilustracija 20: Ugnježdavanje (spajanje) organizacije pod-manifestom

Na prethodnoj ilustraciji prikazan je scenario da su u glavnom manifestu opisana tri čvora tipa *Item*. Dva od njih (*X* i *Z*) sadrže reference (podatak *identifierRef*) na resurse (*X* i *Y*). Čvor *Item Y* složeniji je opisan (povezan) preko pod-manifesta sa čvorovima *Item A* i *Item B*. Time je rešena složenost čvora *Item Y*. Jedan *Item* može da referencira samo jedna resurs. Ugnježdavanjem pod-manifesta, preko čvorova *Item A* i *Item B* ostvareno je obuhvatanje resursa *A* i *B* čvorom *Item Y*. Pored predstavljenog, dozvoljeno je da u okviru jednog *Item* čvora bude više pod čvorova tipa *Item*, takvih da su delimično opisani preko glavnog manifesta, i preko pod-manifesta.

3.2.3.3 IMS Simple Sequencing

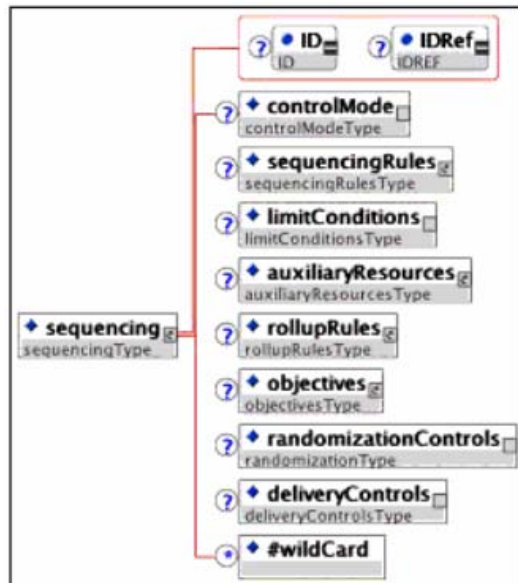
IMS Simple Sequencing specifikacija omogućava da se definiše način na koji će resursi učenja biti pristupačni korisniku (studentu). Specifikacija omogućava da se, u različitim aktivnostima i kontekstima učenja, isti resursi pojavljuju u različitim putanjama [45]. *IMS SS* se zasniva na konceptu aktivnosti učenja (npr. izučavanje nastavnih materijala, odgovaranje na test pitanja). Organizaciona struktura nastave je hijerarhijska – stablo sa čvorovima koji referenciraju odgovarajuće resurse, tako da i aktivnosti imaju stablastu strukturu (Ilustracija 21).



Ilustracija 21: Stablo aktivnosti

Aktivnosti učenja se dekomponuju kroz stablo aktivnosti. Na vrhu hijerarhije je korena aktivnost (npr. pohađanje nekog kursa), dok se najdubljim čvorovima (listovima) predstavljaju konkretne aktivnosti vezane za korišćenje resursa učenja (na slici to su čvorovi označeni brojevima 1-7). U stablo aktivnosti, dodaje se još jedan koncept – klasteri aktivnosti. Klasteri predstavljaju tranzijentne logičke celine susednih čvorova. Klasteri zapravo predstavljaju ogranke stabla. Susedni klasteri mogu se preklapati, a njihova namena je u relaksaciji sistema. Za potrebe navigacije i korišćenja resursa učenja, u sistem se učitavaju samo opisi strukture i resursi obuhvaćeni klasterom.

Da bi se dodala definicija redosleda resursa učenja, specifikacijom je predviđen element *sequencing* (Ilustracija 22) koji se vezuje za *Item* ili *Organization* čvor (definisan *IMS CP* specifikacijom). Na taj način sistem može da prepozna način raspoređivanja resursa koji su uključeni (preko *Item* čvorova) u posmatranu organizaciju.

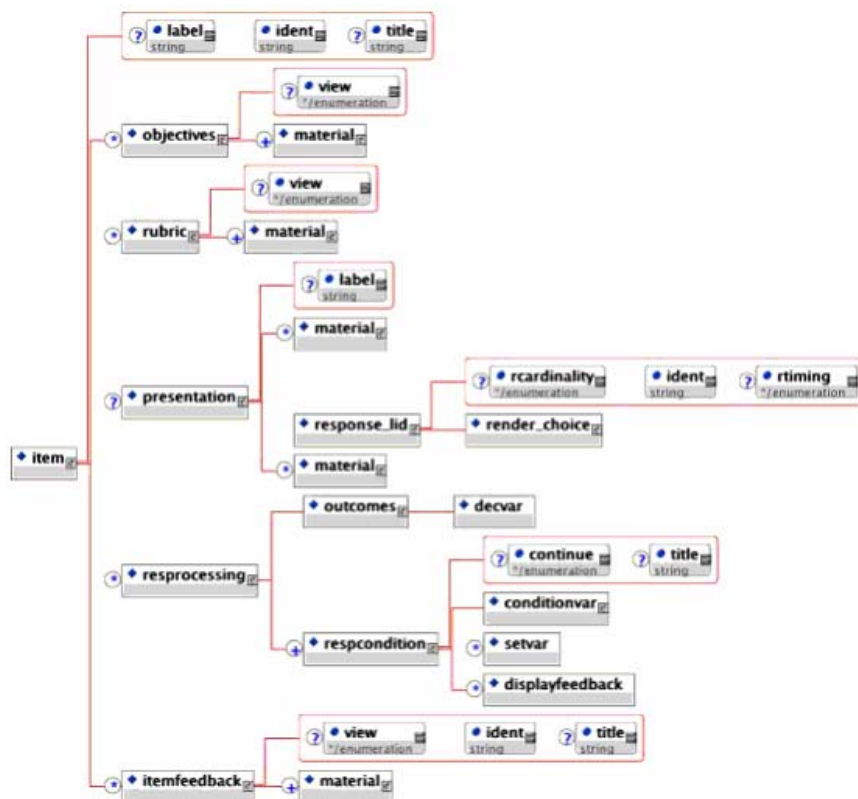


Ilustracija 22: Hijerarhijska sturktura modela aktivnosti učenja

Element *sequencing* se sastoji od 11 atributa. Obavezni su identifikator (*ID*) i referenca na identifikator (*IDRef*). Ovi atributi omogućavaju da se, u posebnom delu manifesta, odvoji kolekcija *sequencing* elemenata, a zatim koristi referenciranjem iz elemenata organizacione strukture (iz *Item* i *Organization* čvorova). *Control mode* specificira način izbora: predefinisanim tokom, automatski napredovanjem, ili slobodnim biranjem. *Sequencing rules* su pravila namenjena za definisanje ponašanja sistema pri raspoređivanju (npr ako je položio pretest, onda preskoči resurs, ako je rezultat slabiji od 0.5, onda ponovi resurs, i sl.). *Limit conditions* da se ograniči broj pokušaja korišćenja resursa. *Auxiliary resources* – za specificiranje pomoćnih resursa (npr. linkovi na rečnike, sistem za pomoć, uputstvo za upotrebu itd.). *Rollup rules* – pravila koja kontrolišu kako se informacije iz pod-aktivnosti reflektuju na čvorove višeg nivoa u stablu aktivnosti. *Objectives* – označava zadatke koji treba da se ispune specificiranom aktivnošću. *Randomization controls* namenjen je definisanju kako će se rešiti redosled pod-aktivnosti posmatranog čvora (aktivnosti). *Delivery controls* se koristi kada je sadržaj isporučen, radi podešavanja ponašanja sistema (npr. podešavanje evidencije – samo zapisivanje, registrovanje kraja pristupanja sadržaju ili urađenog zadatka).

3.2.3.4 IMS Question & Test Interoperability

IMS QTI specifikacija se odnosi na resurse za testiranje. Osnovni koncept specifikacije je *Item* (Ilustracija 23). *Item* odgovara jednom pitanju (zadatku) u testu (proveri). Pored vlastitih atributa (naslov, labela i identifikator), *Item* je struktuiran i sastoji se od pet delova: zadaci, rubrika, prezentacija, procesiranje odgovora i povratna informacija. *IMS QTI* specifikacija podržava veliki broj tipova pitanja (zadataka): jednostruka i višestruka selekcija ponuđenih odgovora, unos odgovora u slobodnoj formi, selekcija regiona (grupe objekata) ili tačke u 2D prostoru (slici).



Ilustracija 23: Schema *IMS QTI* specifikacije

Sledi kratak opis elemenata *Item*-a:

- *Objectives* (zadaci) – opis(i) edukativne namene *Item*-a. Za svaki pogled definiše se drukčiji opis namene (npr. za studenta, nastavnika, inspektora, administratora, ocenjivača).
- *Rubric* (rubrika) – sadrži kontekstualnu informaciju bitnu za *Item* (npr. da li bi trebalo da sadrže neka standardne vrednosti podataka).
- *Presentation* (prezentacija) – sadrži kompletan materijal koji se prezentuje studentu u toku testa (pitanje, odgovori).
- *Resprocessing* (procesiranje odgovora) – uključuje promenjive za vrednovanje odgovora sa ocenama. Pored toga, mogu biti definisani uslovni testovi koji mogu da služe za filtriranje (npr. da li ima barem jedan odgovor, ili da li su označeni svi ponuđeni odgovori).
- *Itemfeedback* (povratna informacija) - povratna informacija koju sistem šalje studentu na osnovu njegovog odgovora. Definiše se za jednu instancu elementa *Resprocessing* (npr. povratna informacija je različita za tačan i netačan odgovor).

```

<questestinterop>
  <item ident="RMr pitanje_1">
    <presentation label="pitanje_1">
      <material>
        <mattext>Token Ring je najrasprostranjeniji tip RMr</mattext>
      </material>
      <response_lid ident="P01" rcardinality="Single" rtiming="No">
        <render_choice>
          <response_label ident="T">
            <material><mattext>Tacno</mattext></material>
          </response_label>
          <response_label ident="F">
            <material><mattext>Netacno</mattext></material>
          </response_label>
        </render_choice>
      </response_lid>
    </presentation>
    <resprocessing>
      <outcomes><decvar/></outcomes>
      <rescondition title="Correct">
        <conditionvar>
          <varequal respident="P01">T</varequal>
        </conditionvar>
        <setvar action="Set">1</setvar>
        <displayfeedback feedbacktype="Response" linkrefid="Correct"/>
      </rescondition>
    </resprocessing>
    <itemfeedback ident="Correct" view="Candidate">
      <material><mattext>Odgovor je tacan.</mattext></material>
    </itemfeedback>
  </item>
</questestinterop>

```

Fragment 4: Primer IMS QTI specifikacije

Na prethodnoj ilustraciji je primer korišćenja ove specifikacije. (Fragment 4). Koreni čvor je *questestinterop* u unutar njega se nalaze pitanja u formi *Item*-a. Prikazan *Item* sadrži tri opciona elementa:

- *presentation* sa pitanjem (crveno uokvireno) i ponuđenim odgovorima (2 odgovora, plavo uokvirena),
- *resprocessing* sa jednim uslovom (Correct), koji u sebi sadrži uslovnu varijablu (ljubičasto uokvirena) povezanu za *response_lid* element *P01* i koja se postavlja na vrednost 1 u slučaju da je odgovor studenta *Tacno* (identifikator *T* za element *response_label*), i
- jednu povratnu informaciju (*itemfeedback*) za slučaj korektnog odgovora (zeleno uokvirena).

Svaki od predstavljenih *IMS* standarda je veoma struktuiran, koncepti su opisani sa velikim brojem atributa, mnoštvom elemenata i podelemenata. Pored hijerarhijske strukture, definisane su i poprečne veze između koncepata. Ove veze se ostvaruju posredstvom referenciranja između elemenata opisa. Splet veza unutar paketa znatno povećava njegovu kompleksnost. Specifikacijom nisu definisani niti granice struktuiranja koncepata (dubina stabla). U praktičnoj primeni, ovaj problem zahteva primenu osetljivih rekurzivnih funkcija za pretragu.

Pored navedenog, nedostatak može da bude i činjenica da ne postoji strategija dodeljivanja identifikatora. Jedino je definisano da unutar paketa sadržaja ne budu koncepti istog tipa i sa istim identifikatorom. Time je odgovornost za rešavanje konflikta imenovanja u slučaju kombinovanja resursa iz različitih paketa na strani korisničkog sistema (aplikacije).

3.2.4 SCORM standard

U novembru 1997.godine (iste godine kad je započet *IMS* projekt), Ministarstvo odbrane (*DoD*) i Odeljenje Bele Kuće za politiku nauke i tehnologije SAD, pokrenulo je Inicijativu za napredno distribuirano učenje (*Advanced Distributed Learning Initiative – ADL*) [118]. Ova organizacija je tvorac *SCORM* (*Sharable Content Object Reference Model*) specifikacije, koja je bila namenjena ostvarivanju sledećih ciljeva: ubrzavanje razvoja različitih dinamičkih i jeftinih sistema i softvera za učenje, i razvoj tržišta za navedene proizvode. Trenutno je aktuelna treća verzija *SCORM*-a (*SCORM 2004 3rd Ed.* iz 2006.godine). Za razliku od prethodno opisanih, ovaj model je mnogo robusniji – pokriva ne samo resurse učenja, već i aspekte organizacije i izvođenja nastavnog procesa. *SCORM* specifikacija je razvijana u odnosu na dva aspekta:

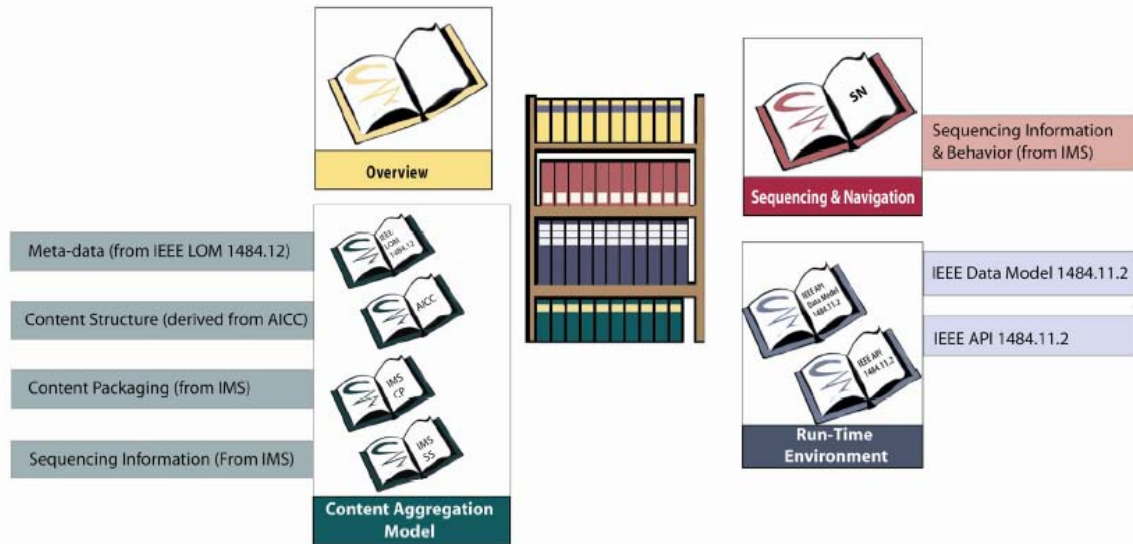
- *Web* kao infrastruktura za učenje – obzirom da *Web* pruža najbolje mogućnosti za maksimiziranje pristupanja i višestrukog korišćenja sadržaja učenja, u *SCORM* specifikaciji se sva razmatranja zasnivaju na Internet mreži, kao medijumu za povezivanje između korisnika resursa i servisa za učenje.
- *LMS* kao davaoci edukativnih servisa - sistemi za upravljanje učenjem (*LMS – Learning Management Systems*) danas predstavljaju najkompletnije aplikacije (besplatne, ili komercijalne), koje pružaju korisnicima različite edukativne servise (davanje, praćenje, izveštavanje i upravljanje sadržajima učenja, praćenje napredovanja studenata, omogućavanje interakcija između korisnika i sistema i delova unutar sistema).

Ovaj standard nije u koliziji sa prethodno opisanim standardima. *SCORM* predstavlja evoluciju, odnosno u sebi sadrži sledeće specifikacije i standarde:

- *IEEE Data Model For Content Object Communication* – model za serijalizaciju i transfer objekata preko javne mreže.
- *IEEE ECMAScript Application Programming Interface for Content to Runtime Services Communication* – specifikacija programatičnog komuniciranja između aplikacija (imalaca servisa) i resursa učenja.
- *IEEE Learning Object Metadata (LOM)* – model za opis resursa učenja (Poglavljje 2.3.2)
- *IEEE Extensible Markup Language (XML) Schema Binding for Learning Object Metadata Data Model* – šema za povezivanje na LOM opisane resurse učenja posredstvom XML jezika i XML šeme.
- *IMS Content Packaging* – standard za pakovanje sadržaja učenja (Poglavljje 2.3.4).

- *IMS Simple Sequencing* – standard za definisanje relacija (redosleda korišćenja) resursa učenja (Poglavlje 2.3.4).

Navedeni standardi i specifikacije su korišćeni za pojedine delove *SCORM*-a. *SCORM* specifikacija se sastoji od četiri velika modula (Ilustracija 24): opšti (pregledni) modul, modul za agregaciju sadržaja, modul za definisanje rasporeda i navigacije po resursima učenja i modul za definisanje okruženja za korišćenje resursa učenja.



Ilustracija 24: Delovi SCORM specifikacije

Sledi kratak opis svakog modula posebno:

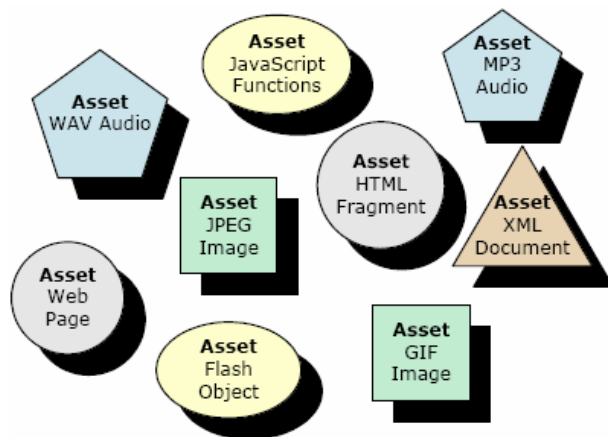
- Opšti pregled (*Overview*) sadrži opšte podatke, koncepte, polazne osnove, istoriju, motive i ciljeve ovog standarda.
- Modul modela za agregaciju sadržaja (*Content Aggregation Model*, skraćeno CAM) – namenjen je za opis (korišćenjem *LOM* specifikacije), struktuiranje (preuzeto iz *AICC*), pakovanje i raspoređivanje resursa učenja (korišćenjem *IMS Content Packaging* specifikacije).
- Modul za definisanje rasporeda i navigacije (*Sequencing&Navigation*, skraćeno SN) – namenjen je za određivanje načina prikazivanja resursa učenja i načina pristupanja resursima od strane studenata kao korisnika sistema (takođe preuzet iz *IMS Simple Sequencing* specifikacije).
- Modul za definisanje okruženja za korišćenje resursa učenja (*Run-Time Environment*, skraćeno RTE) – zadužen je za specifikaciju načina komuniciranja između resursa i aplikacije. Ova komunikacija se odvija preko funkcionalnih poziva definisanih *IEEE API 1484.11.2* specifikacijom, a parametri funkcija i vrednosti koje te funkcije vraćaju su definisani specifikacijom *IEEE Data Model 1484.11.2*.

3.2.4.1 SCORM model agregiranja sadržaja

SCORM CAM specifikacija opisuje način na koji se resursi učenja koriste u izgradnji nastavnih materijala (koji se distribuiraju studentima). Kao što je već rečeno, za opis resursa korišćen je *LOM* standard, a za pakovanje resursa *IMS CP* specifikacija (poglavlja 2.3.2 i 2.3.3). Osnovni koncepti *SCORM CAM* (*Content Agregation Model*) modela su činioči (*assets*), deljivi objekti sadržaja učenja (*Sharable Content Objects – SCOs*), aktivnosti (*activities*), organizacija sadržaja (*Content Organization*) i agregacija sadržaja (*Content Aggregations*). Sledi opisi ovih koncepata, jer njihova agregacija predstavlja *SCORM CAM* specifikaciju.

3.2.4.2 Činioči (Assets)

Činioči su osnovne forme resursa učenja (Ilustracija 25). Na primer, to mogu da budu tekstualni fajlovi, slike, audio fajlovi, objekti za proveravanje znanja, Web stranice, Word, ili PDF dokumenti, XML stranice, ili bilo koji drugi oblik podataka koji može da bude renderovan i predstavljen studentu u Web klijent aplikaciji.

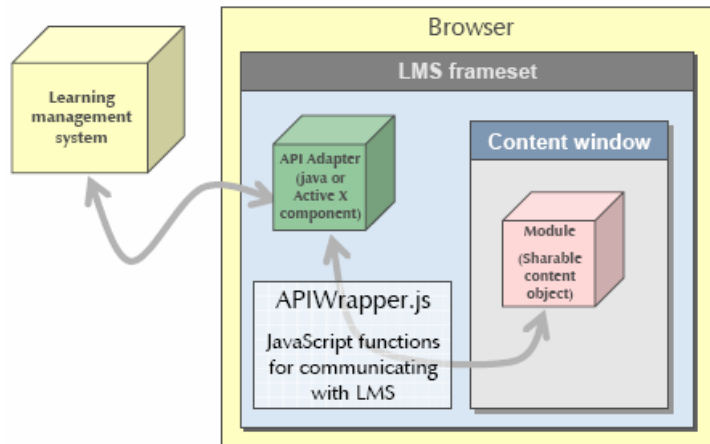


Ilustracija 25: Primeri činilaca (Assets)

Svaki činilac se opisuje metapodacima u cilju omogućavanja pretraživanja i proučavanja unutar repozitorijuma. resursa.

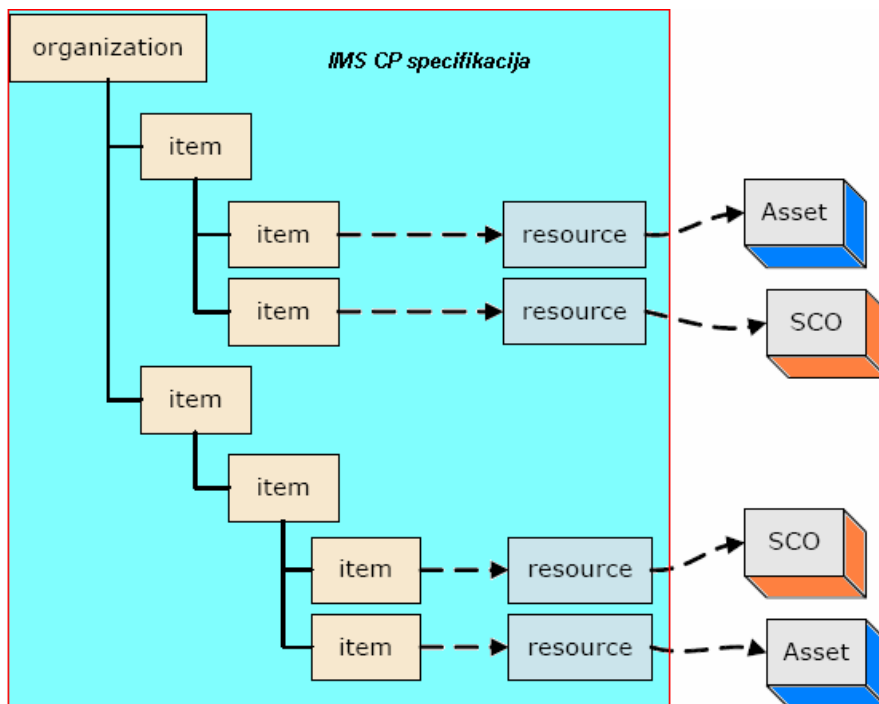
3.2.4.3 Deljivi objekti sadržaja učenja (*Sharable Content Objects – SCOs*)

Deljivi objekti sadržaja učenja (u daljem tekstu označeni kao *SCO*) predstavljaju kolekcije koje sadrže jedan, ili više činilaca, i predstavljaju jedan resurs učenja koji se može samostalno koristiti. Ovo je resurs koji ima najniži nivo granularnosti, a koji se može pratiti od strane sistema za učenje. *SCO* treba da budu male jedinice znanja, kako bi omogućile potencijalnu višestruku upotrebu u više različitih konteksta učenja. Suštinska razlika *SCO* i činilaca je da je *SCO* u stanju da komunicira sa sistemom za učenje korišćenjem standardnog *ECMAScript (IEEE API 1484.11.2)* protokola (Ilustracija 26). To znači da se činioči mogu koristiti kao sadržaj učenja, ali je sadržaj isporučen studentu statički, bez mogućnosti interakcije, izuzev navigacije, ili prelaženja sa stranice na stranicu po unapred utvrđenom redosledu.



Ilustracija 26: Specifičnost SCO komunikacije sa sistemom za učenje

Na prethodnoj slici je opis komunikacije (jedan od scenarija) između sistema za učenje (preciznije, sistema za upravljanje učenjem) i SCO, isporučenog i prikazanog korisniku (studentu) u prozoru pretraživača. Zajedno sa SCO, isporučuje se i JavaScript fajl (označen kao *APIWrapper.js*), i koji sadrži implementaciju standardnih *ECMAScript* funkcija, za uzimanje i vraćanje podataka od/ka sistemu za učenje.



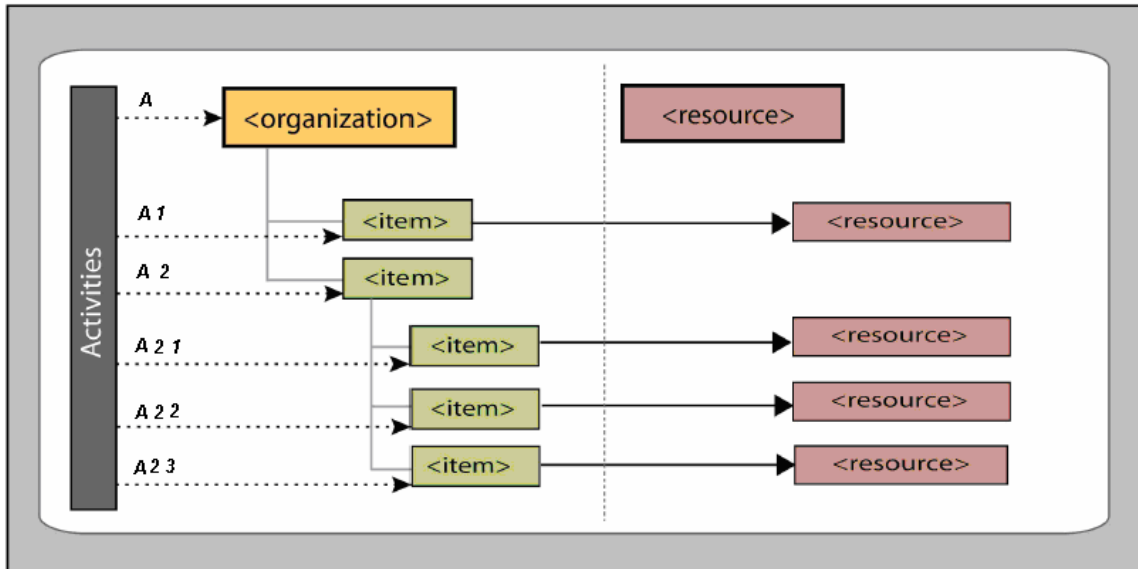
Ilustracija 27: Mesto SCORM koncepta SCO i Assests u IMS CP specifikaciji

Deljivi objekti i činioци sadržaja učenja se dovode u relaciju sa predmetnim činioциma posredstvom koncepta resursa (Ilustracija 27).

3.2.4.4 Aktivnosti (*Activities*)

Koncept aktivnosti u *SCORM*-u kompatibilan je sa istoimenim konceptom, definisanim u *IMS SS* specifikaciji. One omogućavaju raspoređivanje sadržaja (putanja) u cilj

višestrukog korišćenja. Aktivnost učenja se može preciznije opisati kao smislena jedinica nastave. Ona je konceptualno nešto što student obavlja kroz savladavanje edukativnih sadržaja (kroz nastavni proces). Postoje dve vrste aktivnosti: one koje se dalje dekomponuju na pod-aktivnosti, i one koje su direktno povezane za resurse učenja – *činioce* i *SCO* (Ilustracija 28).

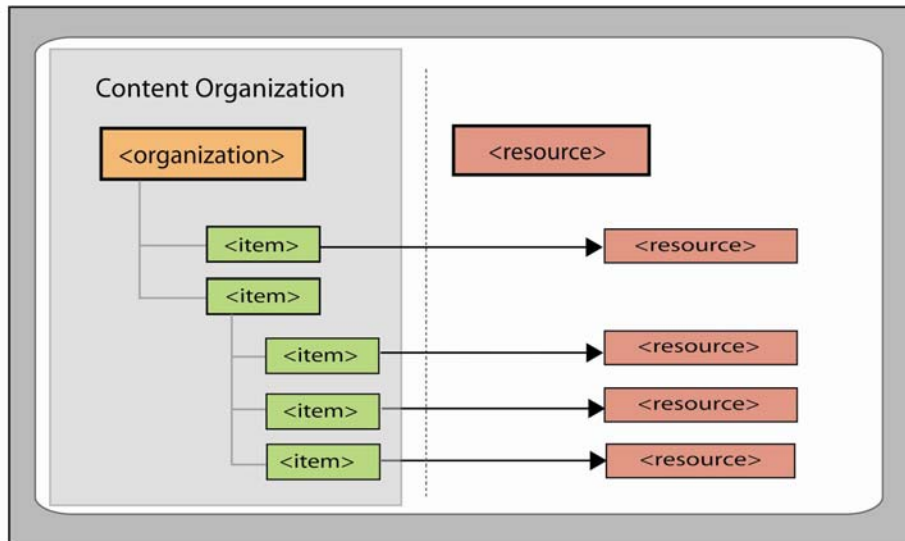


Ilustracija 28: Aktivnosti u SCORM specifikaciji

Na prethodnoj ilustraciji je predstavljen primer relacija između aktivnosti i organizacije sadržaja učenja. Aktivnosti koje sadrže podaktivnosti su A i A2. Aktivnosti koje su povezane sa resursima (sadržajem) su A1, A21, A22 i A23. Aktivnosti povezane sa resursima predstavljaju elementarne aktivnosti. To ne znači da je elementarna aktivnost nestrukturirana i nedeljiva. Na primer, aktivnost učenja nastavnog sadržaja može da se sastoji od tri podaktivnosti: pred-testiranje, proučavanje sadržaja i završno testiranje.

3.2.4.5 Organizacija sadržaja (*Content Organization*)

Organizacija sadržaja je prikaz ili mapa koja definiše namensko korišćenje sadržaja kroz strukturane nastavne jedinice (aktivnosti). Organizacija sadržaja u SCORM-u izvršena je prema *IMS CP* i *SS* specifikaciji. Razlika je u tome što je u *IMS CP* dokumentu, definisanje rasporeda sadržaja (*sequencing&navigation*) – opciono (neobavezno), dok je u SCORM dokumentu mandatorno (obavezno). Organizacija sadržaja je po definiciji hijerarhijska, opisuje se metapodacima i sadrži podatke o rasporedu (redosledu) sadržaja, navigaciji kroz sadržaje učenja. Organizacija je predstavljena konceptima *organization* i *item* (Ilustracija 29). Koncept *organization* u sebi sadrži listu *item*-a. *Item* je koncept koji se povezuje na resurse (posredstvom koncepta *resource*), ili predstavlja strukturu za sebe – sastoji se takođe iz *item*-a. Ovaj slučaj se primenjuje eksplicitno kada je potrebno povezati više resursa na jedan *item*, zbog čega se on dekomponuje (odnos *item-resource* je 1-1).



Ilustracija 29: Hijerarhijska struktura SCORM sadržaja je identična IMS CP

Kao primer korišćenja SCORM specifikacije u agregiranju sadržaja date su taksonomije organizacije nastave u nekim državnim institucijama (organizacijama) za vojnu edukaciju (Tabela 9).

Tabela 9: Organizacija nastave u vojnim obrazovnim institucijama

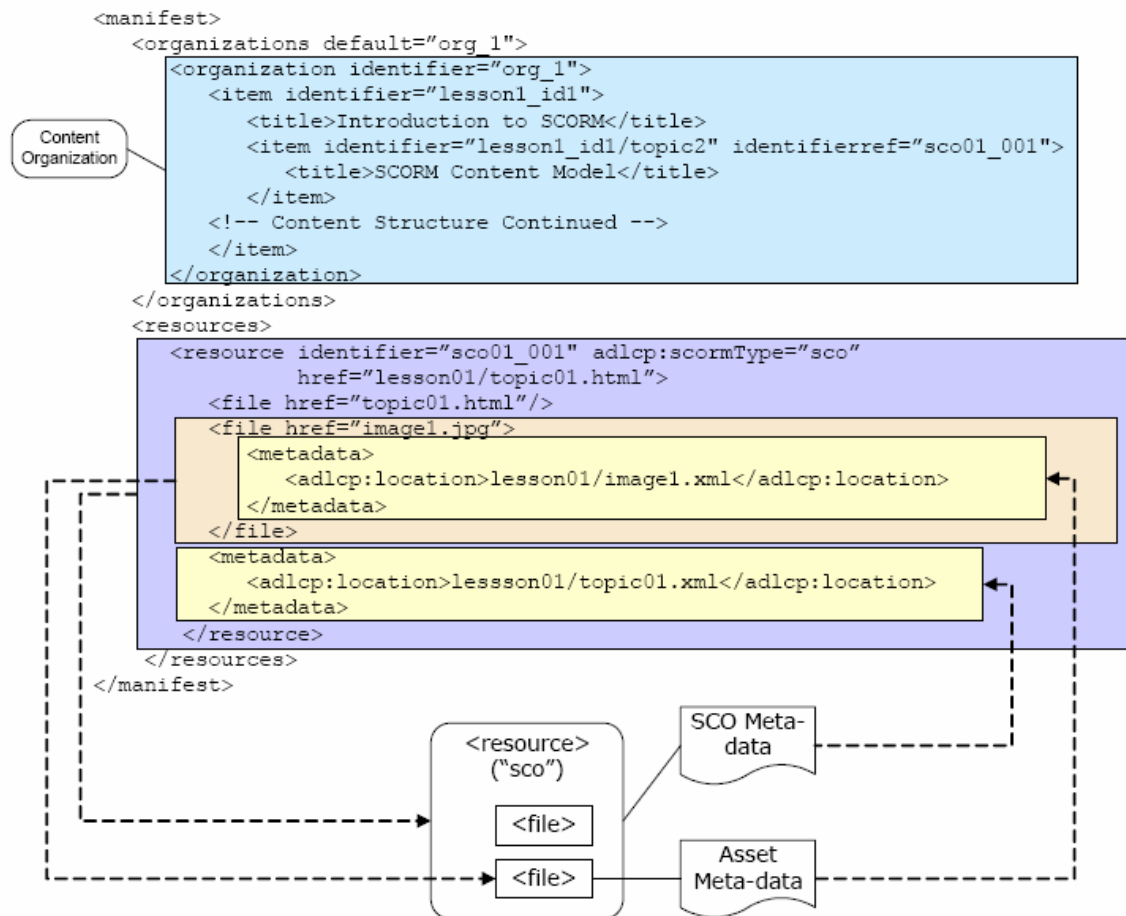
US Army	US Air Force	US Marine Corps	Canadian Armed Forces
Course	Course	Course	Course
Module	Block	Phase	Performance Objective
Lesson	Module	SubCourse (Annex)	Enabling Objective
Learning Objective	Lesson	Lesson	Teaching Point
Learning Step	Learning Objective	Task	
		Learning Objective	
		Learning Step	

Interesantno da postoje razlike u istim obrazovnim sistemima, a koje predstavljaju zaostavštinu samostalnog razvoja. Već na prvom nivou dekompozicije (nulti nivo je nivo kurseva, ili predmeta), postoje razlike u imenovanju. Tematske celine se zovu blokovima, modulima, fazama, ili zadacima performansi. Na drugom nivou se produkuju još izraženije razlike: lekcije, moduli, pod-kursevi (dodaci), zadaci osposobljavanja. Zajedničko za sve sisteme je da se moduli uvek dekomponuju na lekcije, lekcije na konkretne zadatke učenja. Ovaj primer ukazuje da organizacija sadržaja, bez obzira na korišćene standarde za opisivanje sadržaja, može da bude veoma problematična za ostvarivanje interoperabilnosti između različitih edukativnih sistema.

3.2.4.6 Agregacija sadržaja (*Content Aggregations*)

Agregacija sadržaja se koristi za opis akcija i procesa komponovanja skupa funkcionalno povezanih sadržaja učenja. Praktično to je opis paketa sadržaja. Agregacija sadržaja može

da bude korišćena, kako za isporuku sadržaja studentima, tako i isporuku opisa strukture sadržaja i njen prenos između različitih sistema. Princip agregacije je preuzet iz *IMS CP* specifikacije, i jedini novi koncept u agregaciji je dodavanje *SCO* objekata učenja.



Fragment 5: Primer agregacije sadržaja u SCORM dokumentu

Kao što se iz primera može videti (Fragment 5), resursi mogu i dalje da direktno referenciraju na fajlove (primer fajla *topic01.html*), ali je omogućeno da, primenom koncepta *činilaca* (*assets*) i *SCO*, fajlovi budu bolje opisani (primer *činilaca* za fajl *image1.jpg*) i da se grupišu u *SCO* (specifikacija je u fajlu *topic01.xml*). Činioci (*assets*) i deljivi objekti sadržaja učenja (*SCO*) predstavljaju teorijske koncepte koji moraju biti respektovani u toku dizajna i izrade sadržaja učenja. U opisu *SCORM* standarda ne postoje eksplicitni elementi preko koji se ovi koncepti definišu. Oni su implicitno prisutni kao ugnježeni elementi metapodataka namenjeni agregaciji sadržaja u delu koji se tiče resursa (u oblasti *resources*).

SCORM model za opis i agregaciju sadržaja predstavlja samo razradu (operacionalizaciju) prethodno opisanih modela (*LOM* i *IMS* specifikacije). Model je proširen sa dva koncepta, što rezultuje boljom ukupnom agregacionom sposobnošću. Takođe novost je formalizovanje komunikacije između sistema i sadržaja koji se dostavljaju studentima (*ECMAScript* specifikacija). Iako naizgled efemerna proširenja, *SCORM* specifikacija za modelovanje sadržaja je značajna jer je omogućeno znatno povećanje interaktivnosti između sistema i korisnika (pre svega studenata). Značaj ovog

standarda je i u tome što je postignuta integracija različitih standarda (*IMS, LOM, IEEE 1484.11.2, AICC*) i sveobuhvatan pristup u rešavanju problema edukativnih funkcionalnosti i interoperabilnosti između različitih edukativnih sistema.

3.3 Sistemi za elektronsko učenje – kratak istorijat

Sistemi za e-učenje predstavljaju podvrstu edukativnih sistema kod kojih se koristi tehnologija (računari, računarske mreže, pomoćni uređaji) radi ostvarivanja procesa učenja. Ovi sistemi su realizovani kao računarski programi (aplikacije), posredstvom kojih studenti pristupaju različitim resursima učenja. Sistemi za e-učenje su evoluirali uporedo sa razvojem nauke i tehnologije. U narednih nekoliko poglavlja je kratak pregled o tom razvoju.

3.3.1 Generativni sistemi

Šezdesetih godina prošlog veka, istraživači su kreirali brojne sisteme za obuku uz primenu računara (*Computer Added Instructional – CAI*). Ti sistemi su se nazivali *generativnima* [46]. Generisali su skupove problema koji su korišćeni radi povećavanja sposobnosti studenata u domenima baziranim na veštinama (npr. efikasnost u aritmetici, ili prevođenju). Suštinski to su automatizovani sistemi dizajnirani da predstave studentu problem, prihvate odgovor studenta i prikažu celokupne performanse studenta u rešavanju zadatog problema. U to vreme dizajneri su velike napore ulagali u *savladavanje* (programiranje) prilično glomaznih i skupih *mainframe* mašina. Problemi vezani za biheviorističko pitanje *kako* ljudi uče bilo je u drugom planu. Podrazumevalo se da studenti mogu u potpunosti apsorbovati informaciju, koju im sistem prezentuje.

3.3.2 Adaptivno vežbanje i praksa

Kasnih 1960-tih i ranih 1970-tih mnogi istraživači su učinili napredak sa prostog predstavljanja problema studentima i sakupljanja i obrade njihovih odgovora. Počelo se sa razmatranjem *studentskog* faktora [47]. Novi sistemi su promenili tumačenja rezultata studenta. Prvi adaptivni sistemi koji su *modelovali* studenta, koristili su postojeće podatke radi predefinisavanja pattern-a studentskih odziva. Programeri su unapred morali da anticipiraju sve tipove odziva studenata na osnovu kojih bi sistem mogao da odluči o adekvatnoj akciji. Ovi sistemi su samo modelovali ponašanje studenata ali ne i stanje znanja. Iako su bili vrlo jednostavni, u domenu razvoja veština i ponavljanja bili su vrlo efikasni. Studenti su mogli samostalno da izmere svoje sposobnosti i da ih unapređuju. *Implicitna teorija učenja* je podrazumevala da su ovakvi sistemi bili pogodni za sticanje osnovnih sposobnosti i znanja u cilju pripreme za više nivoe obuke.

3.3.3 Kriza edukacione psihologije

Veliki problem napredka u oblastima primene veštačke inteligencije tokom 60-tih, 70-tih i 80-tih, bio je tehnološki razvoj, koji je odlagao apliciranje teorijskih dostignuća za samo deset godina. U to vreme edukaciona psihologija je dovela u pitanje ispravnost teorije biheviorizma. Piaget-ova teorija učenja i konstruktivizam dobijaju na snazi. Chomsky i Newell uveli su ideju simboličnog informacionog procesiranja [48]. Ova ideja se javila sa usmeravanjem pažnje AI zajednice na lingvističko i procesiranje prirodnog jezika. Informaciono procesiranje označeno je kao dominantna paradigma kasnih 1970-

tih i ranih 1980-tih. Ovi radovi su posmatrali ljudsko razmišljanje kao proces u *crnoj kutiji* a ne kao odziv na spoljnu stimulaciju. Nastojalo se da dizajn modela za procesiranje informacija bude pogodan za opis (implementaciju) kroz računarske programe.

3.3.4 Veštačka inteligencija i sistemi za elektronsko učenje

Sleeman i Brown su 1982. godine [49] prikazali stanje u dotadašnjim sistemima za e-učenje i prvi uveli termin „Inteligentni tutorski sistemi“ (*Intelligent Tutoring Systems* – u daljem tekstu *ITS*) radi opisivanja razvijajućih sistema i razlikovanja od prethodnih sistema. Novi sistemi su se razlikovali od svih prethodnih po tome što su imali sposobnost adaptacije - menjanja *ponašanja* prema individualnom studentu, u zavisnosti od studentovih unosa i odgovora. Implicitno se podrazumevalo da se studenti fokusiraju na *učenje radeći*. Oni su klasifikovali ove adaptivne sisteme na:

- monitori u rešavanju problema
- treneri
- laboratorijdk i instruktori
- konsultanti

Iako nisu napuštana istraživanja u vezi primene veštačke inteligencije u procesima zaključivanja o studentu, težište se prebacivalo na istraživanja u vezi predstavljanja znanja studenata u sistemu. Prvi put korišćen je termin *model studenta* kao apstraktne reprezentacije studenta u programu za učenje. Modeli studenta klasifikovani su kao:

- *prekrivajući (overlay)* – studentsko znanje je modelovano kao podskup ekspertskog znanja
- *model razlika (differential)* – sličan je prethodnom, s tim da je fokusiran na razlikama znanja eksperta i studenta.
- *model zabuna (perturbation)* – predstavlja pogrešne koncepte studenta kao varijante proceduralne strukture ispravnih ekspertskih sposobnosti. Veze pogrešnih konceptata studenta i ekspertskog znanja prikazane su kroz *loša pravila (mal-rules)*.

Prva nastojanja u modelovanju studentskog znanja bazirana su na tzv. *buggy* modelu [50]. Ovaj model je zasnovan na studentskim greškama u diskretnim (merljivim) sposobnostima (npr. pogrešan prenos pri oduzimanju). Burton je elaborirao ovaj model sa *DEBUGGY* sistemom. *DEBUGGY* u bazi znaja sadrži 130 grešaka koje se mogu pojaviti u toku oduzimanja. Izazov u modelu je predstavljala analiza problema na osnovu odgovora studenta i određivanje skupa gešaka koje najbolje odražavaju netačno oduzimanje.

Sleeman i Brown su se nadali da će razvoj *ADAPTIVNI SISTEMI ZA ELEKTRONSKO UČENJE* doprineti da teoretičari učenja razviju mnogo preciznije teorije nastave i učenja. Smatrali su da je preciznost moguća i neophodna za implementiranje teorija u edukacioni softver. Smatrali su da je potebno konstruisati okruženje koje bi omogućilo kolaboraciono učenje i da istraživači malo znaju o značaju kooperativnosti u procesu učenja.

3.3.5 Uticaj kognitivnih nauka

Tokom 1980-tih kompjuterske nauke su fokusirale upotrebu veštačke inteligencije oko problema prirodnog jezika, modela studenta i dedukcije. Ovi problemi su često privlačili i naučnike koji se nisu bavili računarima. John Anderson je, baveći se kognitivnim naukama, razvijao teoriju adaptivne kontrole mišljenja (*Adaptive Control of Thought - ACT*) [51].

Iako je Anderson kreirao *ACT* kao kognitivnu teoriju, smatrao je potrebnim da je testira impementiranjem principa u konkretni softver. Dva najpoznatija primera su tutor za geometriju - *Geometry Tutor* (Koedinger & Anderson, 1993) i adaptivni sistem za e-učenje za *LISP* programski jezik - *LISP ITS*. U *LISPITS ACT* principi su implementirani u kontekstu trasiranja modela. Sistem nastoji da modeluje korake u pravljenju *LISP* programa. Program zatim vrši upoređivanje sa aktuelnim kodom koji je student uneo. Corbett i Anderson nazivaju proces praćenja i pomaganja – trasiranjem znanja. Konačni cilj je master model (model savladavanja) u kome svaki student savladava 95% pravila za zadati set vežbi pre nego što pređe na sledeću sekciju. Rezultati istraživanja su bili da studenti koji koriste *LISPITS* brže savladavaju kurs nego studenti koji rade samostalno, ali sporije nego studenti koji su radili sa pravim tutorom.

Andersonovo ime je postalo sinonim za *ITS* isto kao što ljudi pričaju o tutorima Andersonovog stila. Razlog leži u tome što su njegovi sistemi bili zaista korišćeni u učionici i nisu završavali kao istraživački projekti.

3.3.6 Problem esencijalnog modela učenja

Sredinom 1980-tih narastao je entuzijazam u oblasti veštačke inteligencije za kreiranje računara koji misle. Istraživači su se okrenuli ka mnogo prozaičnijim zadacima od pravljenja ekspertnih sistema, koji dobro funkcionišu u ograničenim domenima. To su bili sistemi za dijagnostiku i otklanjanje neispravnosti. Isto tako sistemi za e-učenje su se počeli seliti iz laboratorija u učionice i ostala mesta za obučavanje.

Na njihovu praktičnu primenu u domenu edukacije pojavilo se mnogo raznolikih kritičkih reagovanja. Rosenberg ističe da tekstovi o adaptivnim sistemima za e-učenje sadrže samo nekoliko referenci vezanih za edukacionu literaturu, dok su suviše oslonjeni na materijale iz oblasti računarskih nauka. On je smatrao da ovi nisu zasnovani na esencijalnom modelu učenja. Formulacija modela je učenjena na šturim i nepotpunim analizama. Adaptivni sistemi mogu biti ocenjeni od strane nastavnika i studenata koji ih koriste, ali istraživači se nedovoljno konsultuju sa ekspertima. Testiranja su nepotpuna, neubedljiva i u nekim slučajevima laička. Uglavnom sve pozitivne karakteristike sistema za e-učenje zasnovane su na rezultatima nekoliko obavljenih testova [52].

Tokom istog perioda, konstruktivizam u različitim formama postaje dominantna tema edukacione psihologije. Konstruktivisti su pristupili saznavnim procesima iz mnogo dublje perspektive nego bihevoristi i zastupnici informacionog procesiranja. Saznanje se ne može redukovati na interakciju između velikog broja crnih kutija. Ovakva teorija pada jer ne odražava kognitivnu strategiju studenta. U pet godina (od sinopsis-a Sleeman-a i Brown-a) došlo je do evoluiranja teorijskih postavki adaptivnih sistema za e-učenje. Wenger je predstavio eksplicitne i implicitne ciljeve dizajnera adaptivnih sistema za e-

učenje [53]. On se fokusirao na kognitivne i aspekte učenja, razmatrajući ove sisteme kao komunikaciju znanja.

...razmotrimo ponovo primer knjige:..zamislite aktivnu knjigu koja može interagovati sa čitaocem radi prenošenja znanja na odgovarajućem nivou, selektivno naznačavajući unutrašnje povezanosti i razgranatost stvari, ponovno pozivajući relevantne informacije, ispitujući razumevanje, objašnjavajući teška područja mnogo dublje, preskačući materijale koji su savladani....sistemi sa inteligentnom komunikacijom znanja su ustvari privlačan san.

Wenger poziva da se krene ka kognitivno orjentisanim formama softverskog inženjstva, u kome je težište sa domenskih i pedagoških modela, prebačeno na sazajne procese. Uočio je da od sredine 1980-tih postoje dva različita gledanja na sisteme za e-učenje: tradicionalistički pogled na računare kao uređaje za raspodelu obuke, i pogled na računare kao na alat za istraživačko učenje. Drugin pogled je nov i napredan jer posmatra sisteme za e-učenje kao alat za komunikaciju znanja.

3.3.7 Socijalno-konstruktivistička tranzicija

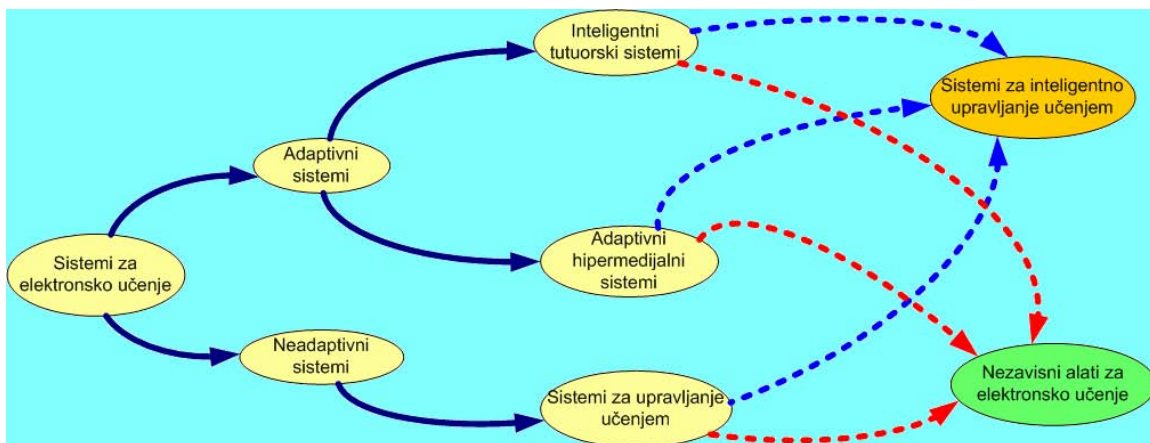
Vygotsky je ispoljio uticaj na edukacionu psihologiju kroz intenzivirano interesovanje za socijalni kontekst. Proučavan je kulturološki uticaj na razvoj individue. Ova ideja je uticala na eminentne istraživače sistema za e-učenje. Početkom 1980-tih na Yale-u Soloway i saradnici su, u okviru projekta *Cognition and Programming Project*, razmatrali probleme okruženja u nastavi iz programiranja koja se izvodi sa novim studentima. Konvencionalna mudrost je bila da programiranje predstavlja podršku u rešavanju problema u ostalim domenima. Soloway i Spohrer tvrde 1989.godine da se saznanja u oblasti rešavanja problema u programiranju mogu iskoristiti u rešavanju problema u potpuno drukčijim domenima. Njihov rad predstavlja istraživanje iz oblasti sistema za e-učenje. Ova dvojica naučnika razvili su širok istraživački program - *HiCE (Highly Interactive Computing Environments)* koji koristi ITS za tutorstvo studenata koji žele da nauče program. Usmereni su bili na modelovanje studenata, kategorizirajući studentske greške, i na kreiranje sistema koji će pomoći studentima da iste identifikuju i da ih otklone (Sack i Soloway, 1992). Sredinom 1990-tih Sack, Soloway i Weingrad su se kritički osvrnuli na vlastiti rad. Smatrali su da se model studenta [54] ne može posmatrati izdvojeno od socijalnog okruženja. Model studenta je po njima deo jedne zajednice znanja. Greška je bila ukalupljavanje modela studenta u model eksperta. Umesto da ga modelujemo, treba da opremimo studenta svim mogućim alatima, sposobnostima i zajednicom koja će mu omogućiti da razvije model prema vlastitim sklonostima i sposobnostima.

3.3.8 Sadašnji trendovi

Internet tehnologija (Web 1.0 i 2.0) je donela pomene koje su prouzrokovale nove načine poimanja stvari (obrazovanja, učenja, socijalnog ponašanja). Student sedeći kod kuće za računarom koji je konektovan na mrežu, iako fizički sam, može da vodi intenzivan socijalan život dopisivanjem, chat-ovanjem, učestvovanjem u diskusionim grupama, na forumima i video-konferencijama. Studenti mogu da se uključe u virtuelne zajednice, shodno svojim interesovanjima.

Nekadašnje pohlepno *skidanje* informacija sa interneta zamenjeno je *postavljanjem* (i čuvanjem) informacija na internetu i korišćenje resursa *on-line*. Praktično ne postoji oblast ljudske delatnosti za koju ne postoje informacije na mreži. Korišćenjem pretraživača, najčešće se do tih informacija brže stiže, nego do informacija koje su smeštene na fajn sistem ličnog računara ili nekim spoljnjim nosiocima (USB Flash, CD, DVD i sl.).

Savremeni sistemi za e-učenje su evoluirali u dva pravca – adaptivni i neadaptivni sistemi (Ilustracija 31). U adaptivne sisteme ubrajaju se dve grupe – inteligentni tutorski sistemi i adaptivni hipermedijalni sistemi. Njihovo zajedničko svojstvo je adaptacija potrebama konkretnog studenta, čime se postiže personalizacija procesa učenja (detaljan opis u poglavlju 3.4). Neadaptivni sistemi su sistemi za upravljanje učenjem (poglavlje 3.5). Ovi sistemi su više fokusirani na sveobuhvatno administriranje edukativnih procesa.



Ilustracija 30: Taksonomija sistema za e-učenje

Tendencija daljeg razvoja sistema za upravljanje učenjem je u konvergenciji aktuelnih sistema tako da se udruže prednosti različitih tipova (adaptivnih i neadaptivnih sistema) u novi kvalitet. Do sada postoji mali broj naučnih radova koji se bave ovim, još uvek nedovoljno istraženim problemom. Postoje dva pravca (pristupa): integracija funkcionalnosti u jednom sistemu i distribuiranje funkcionalnosti u više nezavisnih softverskih alata. U radovima, funkcionalno integrisani sistemi su nazvani “Sistemima za inteligentno upravljanje učenjem” (*Intelligent Learning Management Systems – ILMS*). Nova kategorija sistema imala bi administrativne funkcije kao sistemi za upravljanje učenjem, i sofisticirano ponašanje (funkcionalnosti) koje poseduju adaptivni sistemi.

Drugi pristup zasniva se na činjenici da su postojeći sistemi za upravljanje učenjem previše robusni i veliki deo funkcionalnosti je najčešće u implementacijama samo parcijalno iskorišćen. Takođe, postojanjem standarda za opisivanje i pakovanje resursa učenja, i tehnologija Web servisa, moguće je kreiranje nezavisnih softverskih komponenti koje mogu da budu interoperabilne, lako zamenjive, portabilne i korišćene po potrebi. Ova kategorija softvera je nazvana “Alatima za e-učenje” (*E-learning Tools*).

3.4 Adaptivni sistemi za elektronsko učenje

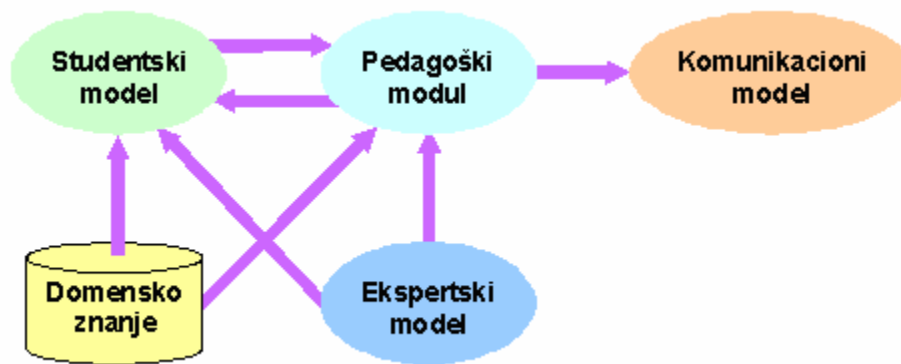
Adaptivni sistemi za e-učenje mogu da se dinamički adaptiraju potrebama individualnog studenta. Tokom interakcije sa sistemom, student unosi podatke, generiše različite događaje, provodi vreme koristeći različite systemske resurse, sarađuje sa drugima kroz

različite aktivnosti učenja. Adaptivni sistemi prate akcije studenata, pamte ih kao podatke u odgovarajućem obliku (u studentski model), donose zaključke o tim podacima (studentima) i na osnovu njih vrše različite tipove adaptacije sadržaja koji se isporučuju studentu. Postoje dve vrste adaptivnih sistema – inteligentni tutorski sistemi i adaptivni edukacioni hipermedijalni sistemi (u daljem tekstu AEHS).

3.4.1 Inteligentni tutorski sistemi

Inteligentni tutorski sistemi (u daljem tekstu *ITS*) su adaptivni sistemi kod kojih je centralna tačka arhitekture – studentski model. Ovi sistemi su domenski usko specijalizovani – namenjeni su za učenje (sticanje) specifičnih znanja i veština. *ITS* ispoljavaju visok stepen interaktivnosti sa korisnikom (studentom), preciznim vođenjem studenta kroz prostor znanja. Česte implementacije sadrže personalizovanog agenta u vidu virtuelnog *tutora*, kako bi se ostvarili bolji psihološki efekti – bliskost između sistema i studenta.

Prvi generalni model *ITS* predstavio je Wenger [53] po kome su glavni činioci u komunikaciji znanja studentski model, domenska i pedagoška ekspertiza, povezivanje znanja u celine potrebne studentu, na osnovu zaključivanja o njegovim performansama, i interfejs posredstvom koga student komunicira sa sistemom. Weber i Brusilowsky [55] su se složili oko konstatacije da "*Tipičan ITS se sastoji od ekspertskog modula, tutorskog modula, modula studenta i interfejsa*", mada nisu specificirali značaj i mesto pojedinih modula u sistemu. Wengerov model je favorizovao ekspertski model (koji modeluje znanje na osnovu zaključaka). Aktuelne arhitekture ističu ravnopravnost modula u *ITS* (Ilustracija 32) [56].



Ilustracija 31: Generička arhitektura inteligentnih tutorskih sistema

Zajedničko za sve savremene *ITS* da obavezno sadrže pet delova: ekspertski model, studentski model, pedagoški modul, domensko znanje i komunikacioni model. Slede opisi ovih delova.

3.4.1.1 Ekspertski modul

U tradicionalnim sistemima za e-učenje, ekspertsko znanje je smešteno u okvire (*frames*) [54]. Sistem prikazuje okvire studentu, dok će redosled okvira biti različit u zavisnosti od odgovora studenta. Ako student tačno odgovori na set pitanja, sistem pokazuje sledeći okvir u nizu, inače biće mu prikazan drukčiji okvir. Okviri su statični i nisu sposobni da

primenjuju znanje koje ima pravi ekspert. Postoje dva pristupa u dizajn eksperskog modula kod savremenih *ITS*.

Prvi (za sada brojniji) zagovara korišćenje sistema produkcionih pravila za konstruisanje ekspertskeg ponašanja sistema. Ovim dobro poznatim pristupom se vrši predikcija ponašanja studenta i pravilima se nastoje obuhvatiti svi mogući slučajevi korišćenja sistema i toka sesije. U sistem se ugrađuju i takozvana sigurnosna pravila koja sprečavaju da dođe do nedefinisanog slučaja korišćenja. Da bi se izbeglo nedefinisano stanje u kome sistem ne može odlučiti između više pravila koje da izvrši uvode se prioriteti pravila (rezolucija rezonovanja). Pravila se grupišu prema funkcionalnosti dok se grupe povezuju posredstvom meta pravila. Na taj način se izbegava aktivacija velikog broja pravila čime se bespotrebno zauzima radna memorija. Pravilski dizajnirana baza znanja se ulančava primenom odgovarajućih strategija u zavisnosti od dubinske pozicioniranosti ciljnih pravila i obima početnog unosa (broja pravila na vrhu lanca ulančavanja).

Drugi (noviji) pristup (ali i druge module u kojima se vrši rezonovanje) implementira ekspertske modul kroz sistem ograničenja. Kod ovih sistema polazi se od ideje da je jednostavnije ograničiti domen. Studentski unosi se ispituju u smislu da li narušavaju neko ograničenje. Narušavanje ograničenja pokreće akciju sistema (alternativni sled događaja). U slučaju da ograničenja u sesiji nisu ugrožena, sistem poprma ponašanje programirane nastave.

U većini komercijalnih sistema autori su se opredelili za potpuno samostalan razvoj ekspertskeg modula bez obzira da li se radi o sistemima pravila ili ograničenja. To znači da se nisu opredeljivali za postojeći ES shell već su timski razvijali vlastiti shell.

Manji *ITS* koji su dela entuzijasta, individualaca oslanjaju se na funkcionalnosti postojećih shell-ova. Ova okolnost danas i nije neko ograničenje, obzirom da postoje vrlo transparentni shell-ovi uz koje se distribuira i izvorni kod, tako da je moguće dograđivanje funkcionalnosti u skladu sa sistemskim zahtevima.

3.4.1.2 Domensko znanje

U novije vreme objavljen je veliki broj radova u vezi predstavljanja domenskog znanja. Tendencija je da materijali za učenje budu modularni i da se kreiraju po opšte prihvaćenim standardima, kako bi bili upotrebljivi u različitim edukativnim sistemima i različitim kursovima (vidi poglavlje 3.2).

Za *ITS* to nije dovoljno zbog njihove domenske specifičnosti. Ovi sistemi imaju vrlo konkretnu namenu i domeni koji se izučavaju su najčešće direktno povezana sa praktičnom primenom znanja. Na primer, u *ITS* kod kojih se učenje odvija rešavanjem problema (*problem based learning*), ne postoji eksplicitno deklarativno znanje, već se to znanje ugrađuje u sistem pravila, ili ograničenja. Kada je prostor znanja mali, a rešenja problema vrlo formalizovana (svedena na set obrazaca za rešavanje), domensko znanje može da bude predstavljeno ograničenjima. Ako prostor znanja ima složeniju strukturu (ne postoji konačni skup obrazaca za rešavanje zadataka), domensko znanje se predstavlja pravilima. Kod *ITS* za učenje rešavanjem problema, pored regularnih modula, postoje i generatori problema (zadatka koji se rešava) i moduli za rešavanje problema – u kojima je takođe implementirano domensko (ekspertske) znanje.

Mesto deklarativnog znanja (obzirom da se radi uglavnom o egzaktnim naukama) u *ITS* je da pruži pomoć u toku rešavanja zadatka, ili da se student pripremi za rad pre dobijanja zadatka. U tom smislu, pored definicija i objašnjenja koncepata, često se koriste primeri (rešeni zadaci).

3.4.1.3 Pedagoški modul

U pedagoškom modulu su implementirane strategije učenja, zapravo načina na koji *ITS* vodi studente kroz savladavanje kursa. Pedagoški modul ima dva aspekta - dijagnostički i didaktički.

Dijagnoza omogućava generisanje studentskog modela na osnovu kog *ITS* adaptira svoje ponašanje prema konkretnom studentu. U procesu dijagnosticiranja sistem treba da meri činioce stanja studentskog znanja, koji su merodavni za rad sistema. Ukoliko je dijagnozom obuhvaćen veći broj činilaca, model će realnije odražavati stanje studenta.

Drugi aspekt pedagoške ekspertize je didaktika. Analogno pravilima, dijagnostika predstavlja premisu, dok didaktička funkcionalnost pedagoškog modula predstavlja njegovu akcionu komponentu. Didaktika obuhvata sve aktivnosti koje *ITS* u procesu učenja preduzima prema studentu na osnovu studentskog modela.

U skladu sa Wengerom tvrdnjom da učenje predstavlja uspešnu tranziciju između stanja znanja, namena nastave (obuke) je usklađena da olakša studentu prelazak u prostor stanja znanja, *ITS* se mora modelovati postojeće znanje studenta i podržati prelazak u novo stanje znanja. *ITS* u tom smislu mora da pruža naizmenično dijagnostičku i didaktičku podršku. U sistemima za e-učenje do informacija o stanju studenta dolazi se kroz tri nivoa:

- na nivou ponašanja, ignorišući znanje studenta i fokusirajući se na ponašanju koje se može opservirati;
- na epistemološkom nivou, u skladu sa znanjem studenta i pokušavajući da zaključi stanje studenta na osnovu posmatranog ponašanja;
- na individualnom nivou, informacije o personalnim karakteristikama studenta (motivacija, lični koncepti u vezi domenske problematike i koncepcije koje student ima prema *ITS*).

Savremeni *ITS* nastoje da fokusiraju dijagnostiku prema spoznaji individualnog nivoa stanja studenta. Profilisanje studenta na nivou individualnih karakteristika je preduslov da bi sistem animirao studenta da bude aktivan učesnik u nastavnom procesu, ne samo pasivan receptor znanja.

Prikupljanje dijagnostičkih informacija predstavlja izazov za dizajnere *ITS*. Za razliku od humane interakcije koja je u perceptivnom smislu relacija ravnopravnih entiteta, receptori računara današnjice su miš i tastatura (vrlo retko mikrofoni ili kamera). Računar može zaključivati o studentu samo na osnovu podataka koje sam student dostavi sistemu (korisničkim unosom). Dijagnostika mora biti vrlo sofisticirana, jer pravi podaci se dobijaju implicitnim metodama. U ljudskoj prirodi je da nema objektivnu predstavu o samom sebi. Eksplicitna i konkretna pitanja tipa da li znaš *LISP* sintaksu verovatno neće pomoći sistemu da dobije realnu sliku o stanju znanja studenta.

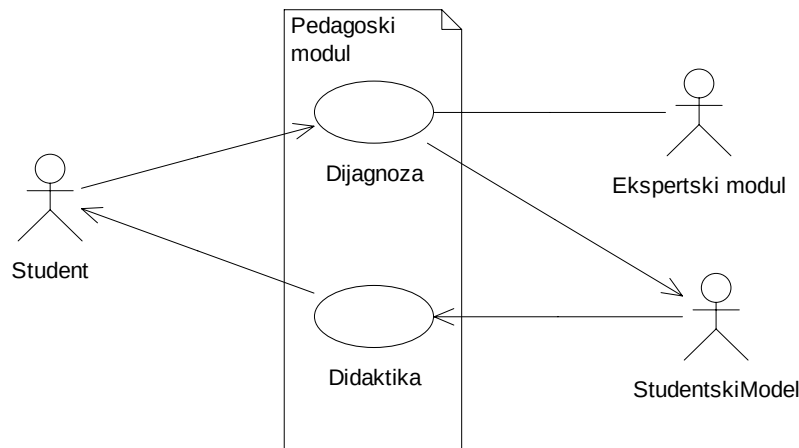
Postoje dva pristupa u korišćenju dijagnostičkih podataka:

- Modelovanje podataka čime sa isti mapiraju u internu reprezentaciju (model) problema. Ovaj pristup je razvijen u programima zasnovanim na okvirima (*frames*). Modelovanjem podataka dizajneri su nastojali da studentovo stanje znanja ukalupe u jedno ili više stanja koja su prepoznatljiva sistemu - u stereotipe.
- Rekonstruktivna interpretacija je zasnovana na modelovanju u kontekstu ciljeva i planova. Svaka sesija započinje sa malim brojem ulaznih podataka. Za razliku od klasifikacije podataka u mali set monolitnih modela, ovaj pristup koristi skupove uslovno nazovimo planova (metode, pravila, ograničenja ili *scheme*). Ovi planovi se dekomponuju u ciljeve i pod-ciljeve. *ITS* u toku prikupljanja podataka filtrira listu radi smanjivanja prostora pretraživanja i određivanja verovatnoća za čvorove prostora pretraživanja. *ITS* verovatnoće određuje dinamički i na taj način sistem reaguje mnogo inteligentnije na tip greške.

Didaktika je drugi aspekt pedagoške ekspertize. Najprostije rečeno, dok dijagnostika predstavlja osnovu za modelovanje studenta, didaktika predstavlja aspekt isporuke u nastavnom procesu. Često su *ITS* više fokusirani na modelovanje i manipulaciju sadržajem ili domenom, dok je didaktika u drugom planu. Modifikacije nastavnih strategija su, kod savremenih *ITS*, vrlo ograničene i u čvrstoj su sprezi sa specifičnošću domena. U komparativnoj analizi *ITS* (tabela 1) može se uočiti da je u svakom sistemu zastupljena (ili dominantna) samo jedna od strategija učenja (uglavnom je to aktivno učenje rešavanjem problema uz korišćenje kolaborativnih servisa da bi se, pored individualnog, omogućio koncept grupnog rada). Nastavnička praksa je pokazala da sistemi treba da implementiraju različite strategije, a modalitete njihovog korišćenja sistem treba da određuje na osnovu studentskog modela.

Didaktika se može organizovati oko četiri principa koji mogu biti važni za budućnost *ITS* istraživanja (Wenger):

- Planovi akcije: "Mini nastavni plan" koji se koristi poput koridora da vodi studenta i obezbeđuje kontekst za dijagnostičke operacije.
- Strategijski kontekst: u kome se implementiraju planovi akcije. Na strategijskom nivou kontekst može biti unapred zadat ili oportunistički - sa mogućnošću modifikovanja u toku studentske sesije.
- Baza odlučivanja: pravila ili ograničenja za alociranje sistemskih resursa - granica korišćenja sistema.
- Ciljni nivo studentskog modela: u bilo kom momentu student može biti na nivou ponašanja, konceptualnom nivou ili meta-kognitivnom nivou. Može istovremeno biti na više nivoa. Realan nastavnik može razumevati i odgovarati na odgovarajući način na svaki od nivoa. Do sada, naj veći broji *ITS* fokusirani su na nivo ponašanja, ili pokušavaju da zaključie konceptualni nivo studenta na osnovu ponašanja. Retki su sistemi koji pokušavaju da determinišu na kom nivou je studentu potrebna pomoć u datom momentu i sistemi koji određuju usklađen plan akcija.



Ilustracija 32: Opis pedagoškog modula pomoću dijagrama slučajeva korišćenja

Da bi rezimirali razmatranja o pedagoškom modulu opisali smo ga korišćenjem UML alata (Ilustracija 33). Mogu se uočiti relacije i vidljivost između pojedinih modula i aktera. Na osnovu podataka koje student unosi u sistem i konsultujući pri tome ekspertski modul (za zaključivanje), pedagoški modul omogućava modelovanje stanja studenta. Studentski model zatim koristi za didaktičke aktivnosti prema studentu. Ovaj ciklični tok događanja se ponavlja sve vreme sesije studenta sa sistemom.

3.4.1.4 Studentski model

Modelovanje studenta je srž *ITS* (Holt, Dubs, Jones & Greer, 1994). Razlika *ITS* od ostalih sistema za e-učenje u sposobnosti sistema da odgovori na individualni stil učenja studenta i izvrši raspodelu prilagođenih instrukcija (studentskom modelu). Iako je studentski model specijalizacija korisničkih modela, naučno dobro opisanih (vidi poglavlje 2.2), i postoje brojni standardi za modelovanje studenta (vidi poglavlje 3.1), i dalje je model studenta u *ITS* privlačna tema za istraživanje. Specifičnost primene *ITS* uslovia je potrebu dizajna specifičnih modela. *ITS* treba da modeluje znanje, studenta, kao i odnos nastavnik - student (Mitchell & Grogono, 1993). Bez obzira na kompleksnost *ITS*, neadekvatna adaptacija sistema u odnosu na stanje studenta, dovešće do ograničene praktične primene sistema. Model studenta u *ITS* ima tri zadatka (Wenger):

- Prikupljanje podataka o studentu. Ti podaci mogu biti eksplicitni - zahtevajući od studenta da reši određene probleme, ili implicitni - prateći navigaciju studenta, i ostale interakcije sa sistemom. Implicitni podaci se zatim upoređuju sa sličnim pattern-ima ponašanja.
- Korišćenje prikupljenih podataka radi kreiranja predstave o znanju studenta i procesu njegovog učenja. Često je to otvoren model koji predstavlja znanje studenta u vidu devijacije od znanja eksperta. Sistem koristi takav model radi predviđanja budućih akcija (odziva) studenta. Povratnom spregom sistem poredi predviđane sa stvarnim odzivima i u inkrementalno-iterativnoj petlji poboljšava model studenta tokom čitave sesije.
- Analitičko procesiranje podataka u cilju izvršenja neke vrste dijagnostike - stanja znanja studenta i radi izbora optimalne pedagoške strategije u izvođenju nastave

koja sledi. U nastavnoj praksi, studenti čije je znanje krhko i daju nesigurne odgovore, nemaju konzistentan odziv. Veliki izazov predstavlja čišćenje šuma, odnosno izbacivanje nekonzistentnih podataka pre analitičke obrade.

Dizajn studentskog modela u ITS se razlikuje od sistema do sistema. Generalno studentski model je prilagođen nameni, implementaciji i konkretnim zahtevima kojima ITS treba da odgovori. Postoje mnogobrojni implementacioni primeri. U sistemu MT-SAT [119], za modelovanje studenta korišćene su Bajesove mreže (Bayesian Networks), dok se u nekim ITS alatima koji su pomoć studentu u toku procesa učenja zasnovali na agentima (*Merlin agent* - alat za pomoć u datom kontekstu i za podršku kolaborativnom učenju) koriste Kohonen-ove mape za samoorganizovanje (Self-Organizing Map SOM).

Jedan od problema kome se poklanja naročita pažnja je inicijalizacija modela studenta (*Initializing Student Model - ISM*). U njegovom rešavanju postoji niz različitih pristupa. Najjednostavniji pristup je da student ne zna ništa o domenu (*absolute beginner*). U ovom slučaju prva postavka sistema može biti neefikasna. Da bi se dobio psihološki profil novog studenta, sistem nameće upitnik (*questionary*) u koji student unosi podatke o sebi. Na ovaj način otkrivaju se sklonosti, interesovanja, ciljevi i motivacija studenta. Da bi se procenio nivo znanja, primenjuju se različite forme testiranja (*pretest*). Pomoću upitnika i testova, upoređivanjem sa očekivanim miljeima (patterns) formira se inicijalni profil individualnog studenta.

Kod pojedinih ITS ide se korak dalje, pa se na osnovu inicijalnog studentskog modela generiše sekundarni model, korišćenjem odgovarajućih algoritama (npr. *Distance Weighted K-nearest Agorithm*). Algoritam ima funkciju da izmeri odstupanje snimljenih karakteristika studenta u odnosu na neke "nominalne" vrednosti. Uprošćeno, pored znanja meri se i veličina greške, jer će ona uticati na efekte studentovog učenja u pojedinim segmentima (fazama) nastave. Primeri sistema koji imaju ugrađen *ISM framework* su *Web-PVT* (Web Passive Voice Tutor) [120] i *Web-EasyMath* [121].

3.4.1.5 Komunikacioni model

Ovaj model je odgovoran za komunikaciju između studenta i ostalih komponenti ITS. On predstavlja sponu posredstvom koje se obavlja transfer znanja između sistema i studenta. Sistem će studentu pružiti ugrađeno ekspertsko znanje, dok će kroz odzive, sistem učiti o studentu. Sistem će dalje, pokušati da olakša učenje studentu.

U novije vreme sve veća pažnja se poklanja psiho-socijalnim istraživanjima i dostignućima u dizajnu korisničkog interfejsa. Performanse današnjih računara omogućavaju dizajnerima interfejsa ispoljavanje svakog vida kreativnosti bez ikakvih ograničenja u kompleksnosti, ili multimedijalnosti predstavljenih sadržaja.

Moraju se međutim uzeti u obzir i pedagoške implikacije interfejsa ITS. Wenger sugerise da interfejs mora da sadrži model koji će rešavati dvosmislenosti u studentskom odzivu. Poteškoću predstavljaju kontradiktorni i/ili nepotpuni odzivi studenta. Za uspešan dijagnostički proces vrlo je važno da odzivi budu ispravni i upotrebljivi. Na taj način ITS može da eliminiše bespotrebna ponavljanja u procesu učenja i da se fokusira na zaista potrebne instrukcije. Na primer sistem *SOPHIE* [122] je eksplicitno dizajniran za pretraživanje vlastite baze znanja radi upoređivanja sa studentskim odgovorima radi dobijanja približnog poklapanja u nastojanju oporavka od neočekivanih odziva studenta.

Drugi problem interfejsa je predstavljanje znanja. Ako sistem učini znanje u potpunosti dostupnim on postaje okruženje za istraživanje znanja. Na taj način sva odgovornost u učenju prenesena je na studenta, koji mora da pretražuje materijale korišćenjem interfejsa. Takva je situacija na Web-u. Problemi nastaju kada materijala ima mnogo i kod studenta se stvara informaciono zagušenje i odbojnost prema učenju. Da bi se uspostavila komunikacija znanja ITS mora da obezbedi neku vrstu vođenja (čak treniranja) radi prevencije da student ne ispusti neki važan aspekt domena. Želja je da ITS obezbede mnogo aktivnije vođenje ili tutorstvo, jer povećava spektar mogućnosti u kojima ITS mogu zameniti prave nastavnike.

U dizajnu interfejsa, kao i na svakom drugom polju gde se ispoljava smisao za estetiku, mora se voditi računa o odmerenosti u korišćenim bojama, ilustracijama i uopšte, svim multimedijalnim sadržajima, skladu sa celinom sistema. Isto tako, preterano nastojanje da sistem podražava ljudska svojstva (kao što se razvija priča o personalizovanim agentima, najčešće asistentima), može da odvlači pažnju umesto da inspiriše. Treba biti svestan činjenice da niko normalan neće doživeti komunikaciju sa mašinom kao komunikaciju sa stvarnim ljudskim bićem.

3.4.2 Primeri inteligentnih tutorskih sistema

3.4.2.1 Cognitive Tutor

PAT OnLine (Ritter, 1997) je jedan od prvih primera ITS zasnovanih na Web-u [57]. To je bio tutor za algebru koji vrši trasiranje modela. *PAT OnLine* je široko prihvaćen u SAD i preporučen od strane Nacionalnog saveta nastavnika matematike (*NCTM*, 1989), a zatim je evoluirao u mnogo složeniji sistem - *Cognitive Tutor* [58]. Ova nova i domenski proširena verzija se masovno distribuira i pokriva mnogo više matematičkih oblasti od *PAT Online* sistema (aritmetika 1 i 2, geometrija itd.). Distribucioni paket je predviđen za rad u lokalnim mrežama edukativnih institucija, iako omogućava internet pristup kabinetskom serveru sa udaljenog računara (doduše posredstvom 1MB i više brze mreže).

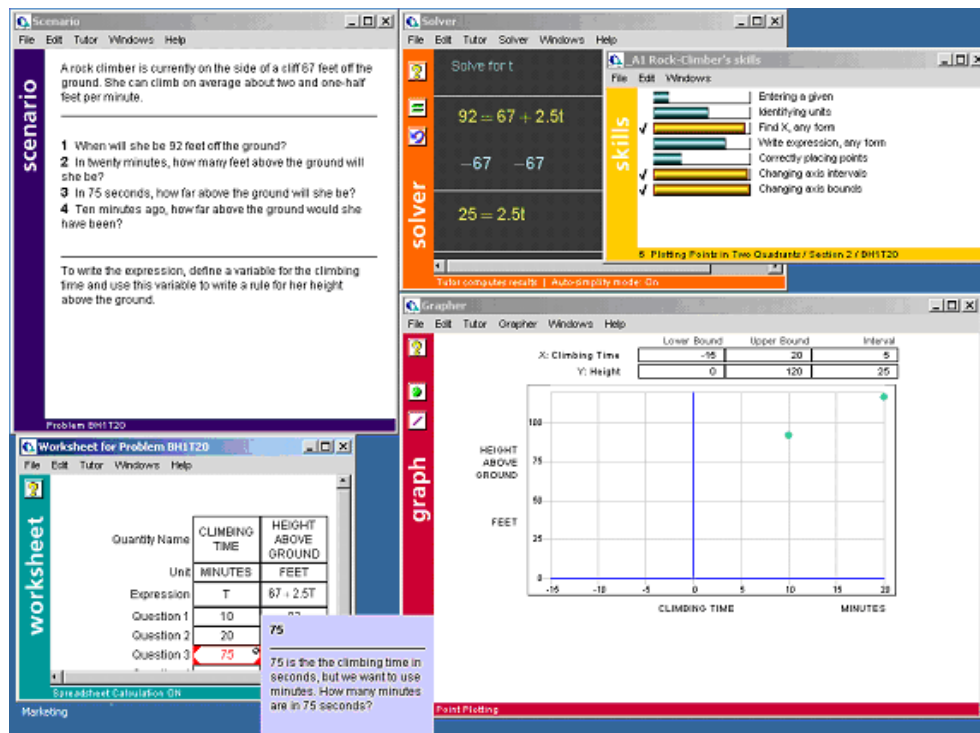
Cognitive Tutor se koristi korišćenjem standardnog Web pretraživača. Nakon što se registruje u sistem, korisnik bira nivo i temu koju treba da savlada. Svaka tema je izdvojena na lekcije. Pored sadržaja koji se mogu proučavati, za svaku lekciju postoji test. Ne postoji ograničenje da se test može realizovati pre nego što se ne prođe materijal lekcije. Nakon izvršenog testa sistem daje obrazloženje sa ukazivanjem na grešku u rešavanju.

Cognitive Tutor je baziran na kognitivnom modelu, koji simulira razmišljanje studenta. Sistem vodi studenta kroz procese trasiranja modela i trasiranja znanja. Određivanje studentskog modela sistem vrši odgovaranjem na svaku individualnu strategiju učenja studenta. Trasiranje znanja (Anderson, Boyle, Corbett, i Lewis, 1990) je određivanje u kojoj meri student vlada znanjima iz oblasti izabrane za izučavanje. Sistem omogućava učenje rešavanjem konkretnih matematičkih problema (Ilustracija 34).

Cognitive Tutor sadrži ekspertski sistem koji je sposoban da rešava probleme koji se šalju studentima. Sistem prati rešavanje problema od strane studenta i vrši upoređivanje sa vlastitim koracima u rešavanju problema. Ako razlike nema, znači da je student uspešno rešio problem. U slučaju da *Cognitive Tutor* ne prepozna akcije studenta kao jednu od

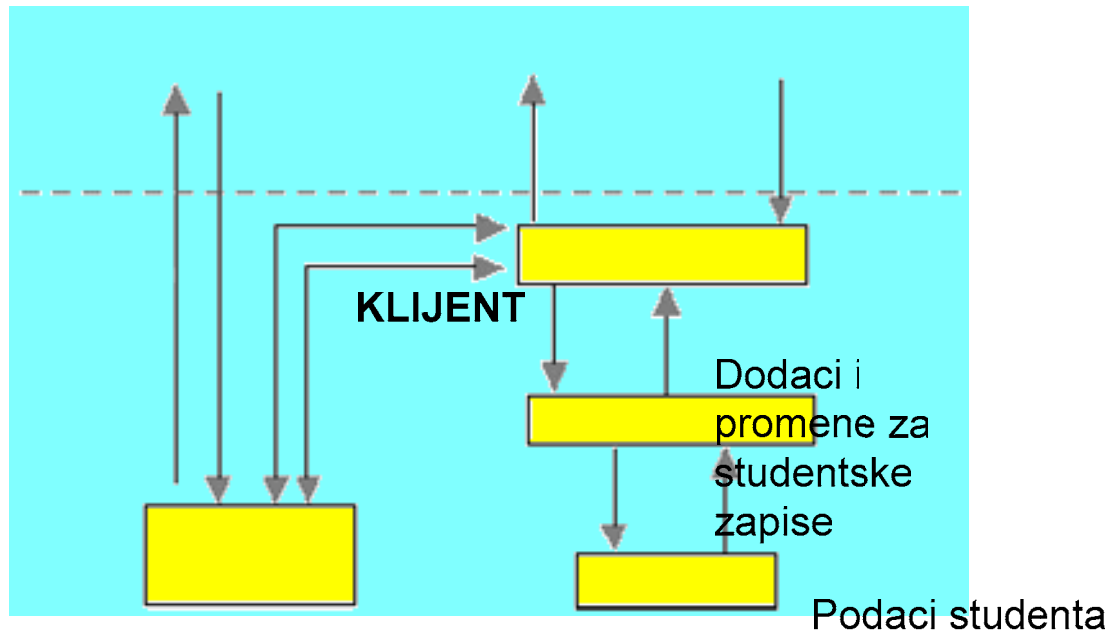
moću putanja u rešavanju problema, upoređuje ih sa uobičajenim propustima. Ako Cognitive Tutor prepozna grešku, sposoban je da obezbedi instrukcije kako bi student prevazišao grešku.

Ekspertski sistem unutar tutora je predstavljen kao kognitivni model. Svako pravilo ovog modela predstavlja pojedinačnu komponentu u sposobnosti sistema da reši problem. Na taj način pravila mogu biti mapirana na kognitivne sposobnosti isto kao i akcije studenta. Svaki studentov korak u rešavanju problema predstavlja elementarnu informaciju o studentovom znanju. Anderson i Corbett (1992) opisuju specifičan algoritam koji tutoru može omogućiti merljivost znanja na osnovu akcija studenta. Ovaj algoritam omogućava specifičan način ocenjivanja studenta. Znanje studenta se meri relativno u odnosu na njegove fundamentalne (kognitivne) sposobnosti, za razliku od komparacije sa apsolutnim standardima. Student uvek ima prisutan grafički prikaz svog znanja iz odabrane teme preko napunjenosti linija (menzure) koje se nalaze pored naziva svake lekcije. Ovaj graf je popularno nazvan skillometer. U osnovi ove ideje da prikaže kako učenje učestvuje u povećavanju kognitivnih sposobnosti studenta. Tutor meri napredovanje studenta u odnosu na studentove osnovne sposobnosti.



Ilustracija 33: Studentski interfejs *Cognitive Tutor-a*

Tutorski agent predstavlja jezgro *Cognitive Tutor-a*. Na osnovu dijagrama arhitekture sistema može se zaključiti da je sva logika smeštena na serveru (Ilustracija 35).



Ilustracija 34: Arhitektura Cognitivnog Tutora

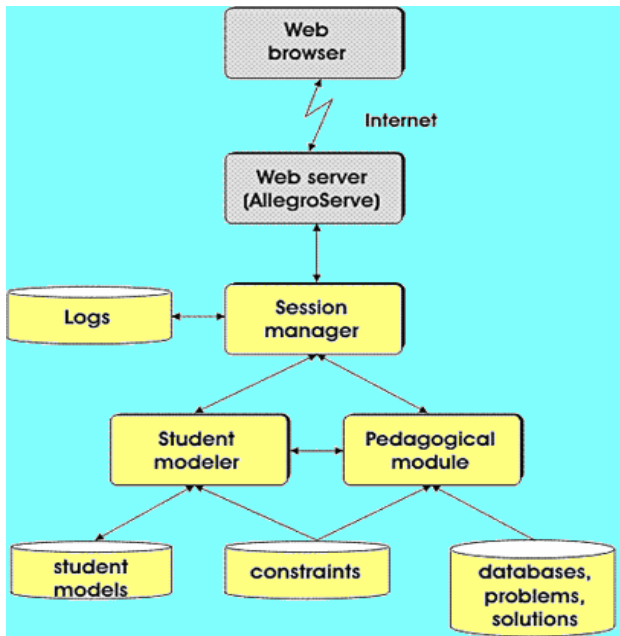
Za generisanje HTML forme sa sesalju na klijentski računar koristi se CGI (Common Gateway Interface) program. Translator prevodi korisnički (studentski) unos preko HTML forme u format koji tutorski agent može da obradi. Studentski model se čuva u serveru studentskih zapisa. Ovakvo arhitekturno rešenje je proisteklo iz nastojanja da se iskoriste klase prethodne verzije sistema (pre svega tutorskog agenta).

3.4.2.2 SQL Tutor

SQL Tutor je ITS za učenje SQL jezika za rad sa bazama podataka [59] i [60]. Adaptivnost sistema ispoljena je kroz podešavanje nivoa kompleksnosti problema prema individualnom studentu. Pored izučavanja nastavnih materijala, studenti rešavaju problemske zadatke. Sistem obezbeđuje povratne informacije koje pružaju pomoć studentu u slučaju grešaka i prilagođava sledeće nastavne zadatke na osnovu odziva.

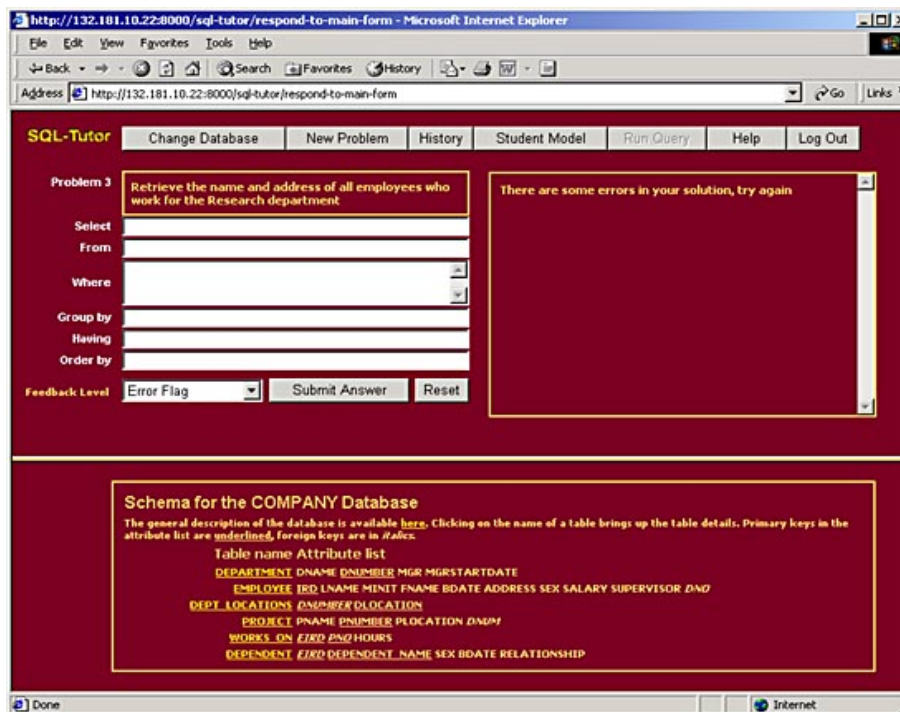
Postoji više verzija ovog sistema (za različite platforme) od kojih je interesantna Web verzija. SQL Tutor ima arhitekturu sličnu većini ITS (Ilustracija 36): pedagoški modul, studentski modelar, vrlo kompleksnu i distribuiranu bazu znanja, menadžera sesije, Web komponente kao interfejs (komunikacioni modul).

Jedinstvenost ovog sistema čini način modelovanja studenta. Bazu znanja pored lekcija, problema i njihovih rešenja čine i ograničenja (*Constraints*). Ograničenja su fokusirana na ugrožavanje osnovnih principa domenskog znanja. Sistem ne meri znanje studenta, već utvrđuje stepen odstupanja rešenja koje je dao student od idealnog rešenja za dati problem. Modelovanje na osnovu ograničenja (*Constraint-Based Modeling - CBM*) je pristup koji je ustanovio Stellan Ohlsson. Sistem trenutno sadrži 800 ograničenja koja se proveravaju kroz *RETE* mrežu (*Rete* algoritam). Za svakog studenta pravi se personalizovani model na osnovu kojeg sistem prilagođava kompleksnost i tempo savladavanja nastave.



Ilustracija 35: Arhitektura SQL Tutor-a

Studentski model u *SQL Tutor*-u sadrži podatke o prethodnim sesijama i podatke o znanju studenta (znanje studenta je takođe predstavljeno ograničenjima). Student u sesiji bira tip DBMS i vrstu problema kojim želi da se bavi u toku sesije (slika 12). Iz korisničkog ugla ovo je jedan od sistema kod kojih je učenje zasnovano na rešavanju problema.



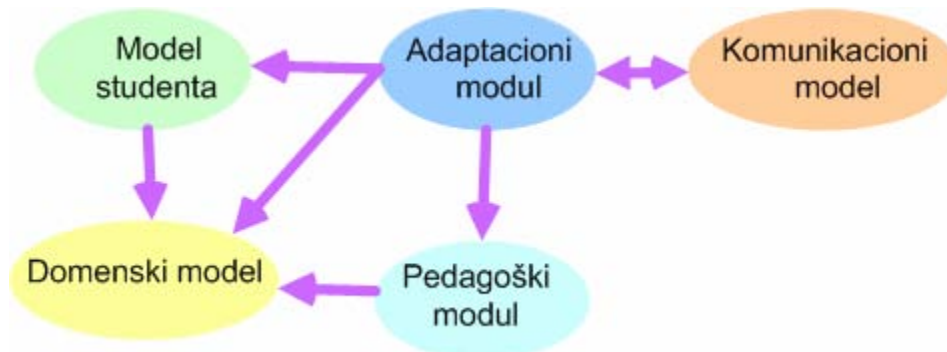
Ilustracija 36: Interfejs Web verzije SQL Tutor-a

Za izučavanje domena baza podataka koriste se slični sistemi koji su se razvili diverzifikacijom *SQL Tutor*-a: *KERMIT* (*Knowledge-based Entity-Relationship Modelling Intelligent Tutoring System*) [62], *NORMIT* (*NORMALisation Intelligent Tutor*) [61], ali i sistem *CAPIT* za učenje gramatike engleskog jezika (*Capitalisation And Punctuation Intelligent Tutor*) [63]. Svaki od ovih sistema sadrži ključne delove baze znanja i studentskog modela zasnovane na ograničenjima.

Iako su sva tri sistema rezidentna na istom mestu (*University of Canterbury*), ne postoji nikakva povezanost među njima. Pedagoška strategija se zasniva isključivo na rešavanju problema. Materijali za učenje su obrađeni u meri u kojoj pružaju dovoljno predznanja za rešavanje problema. Sva tri sistema su gotov softverski proizvod koji su komercijalno gledano, u potpunosti odgovorili nameni.

3.4.3 Adaptivni edukativni hipermedijalni sistemi

Adaptivni edukativni hipermedijalni sistemi (u daljem tekstu *AEH* sistemi) koji su fokusirani na sadržaje učenja. Postoje dve ključne razlike u odnosu na *ITS* sisteme. *AEH* sistemi sadrže kao i *ITS* studentski model (Ilustracija 38), ali je, umesto pedagoške ekspertize studenta, adaptacija sadržaja suštinska funkcionalnost *AEH* sistema. Druga razlika je u modelovanju domenskog znanja. Kod *ITS*, domensko znanje se definiše skupovima pravila i/ili ograničenja, dok je domensko znanje kod *AEH* sistema predstavljeno eksplicitno – hipermedijalnim nastavnim materijalima.



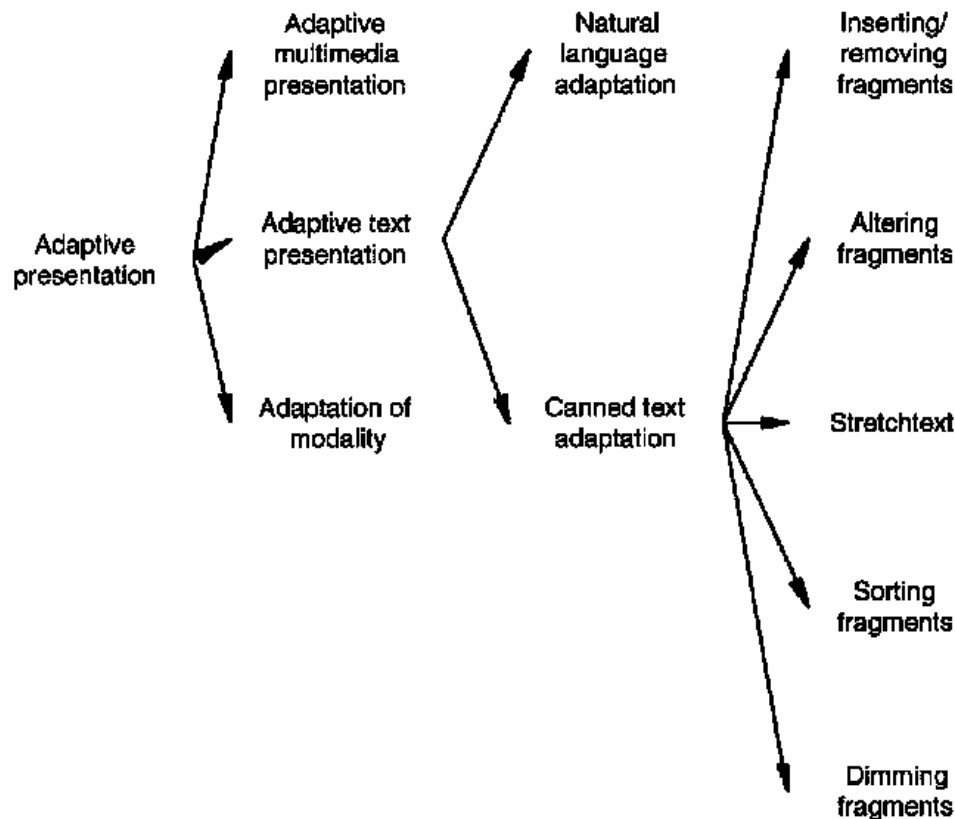
Ilustracija 37: Arhitektura adaptivnog edukativnog hipermedijalnog sistema

Adaptacija sadržaja odvoja se u adaptacionom modulu. Ovaj modul koristi pedagogiju (pedagoški modul) primenjenu na domenski model, rezultate i status studenta (destilovane iz studentskog modela), da bi adaptirao sadržaje koji se isporučuju studentu preko komunikacionog modela. U *AEH* sistemima postoje dve osnovne adaptacione tehnike [123]:

- Adaptacija predstavljanja sadržaja
- Adaptacija navigacije u sadržaju

3.4.3.1 Adaptacija predstavljanja sadržaja

Mogućnosti za adaptaciju predstavljanja sadržaja zavise od stepena fragmentiranosti i deskripcije sadržaja učenja i od tipa sadržaja koji se adaptira (multimedijalni sadržaji, ili tekstovi). Adaptacija tekstualnih sadržaja (Ilustracija 39) odnosi se na jezičku lokalizaciju (promenu jezika, korišćenje nacionalnih alfabeti), i na primenu različitih načina predstavljanja teksta u konkretnom kontekstu učenja (da li se tekst prikazuje kao dodat, ili alternativni fragment, ili zauzima čitavu formu/stranicu, da li se radi o sortiranju tekstualnih fragmenata, ili se fragmenti skrivaju i prikazuju po potrebi).

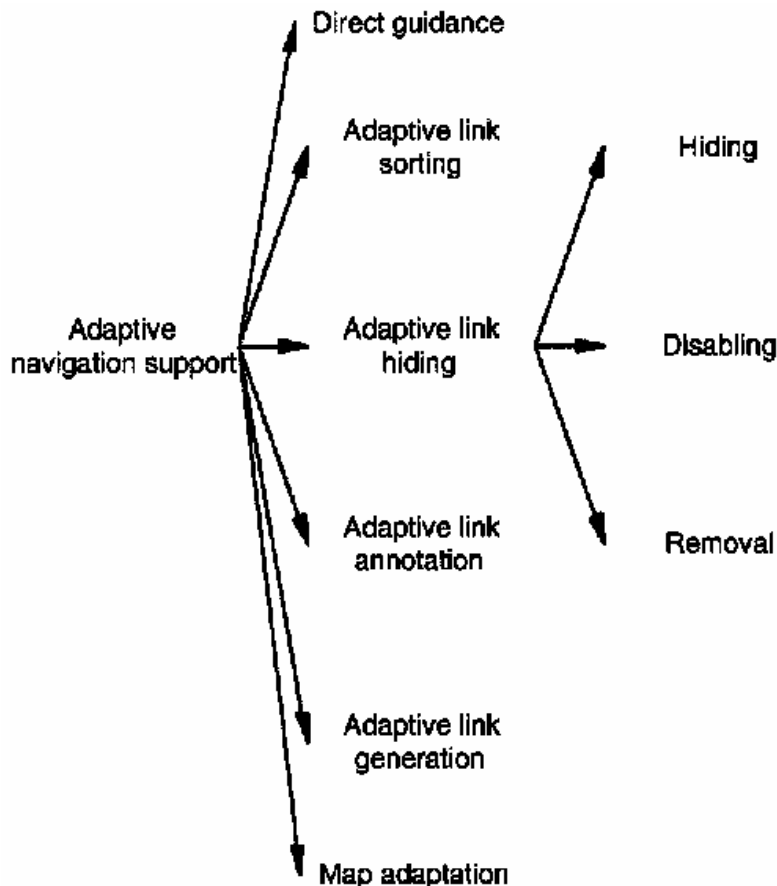


Ilustracija 38: Taksonomija adaptacije predstavljanja sadržaja

U predstavljenoj taksonomiji nisu predstavljene kategorije za multimedijalne sadržaje i modalitete (kombinacije teksta sa drugim tipovima sadržaja). Adaptacija ilustracija (slika) može da se razmatra slično tekstualnoj adaptaciji – prikazivanje linka na sliku, prikazivanje u različitim dimenzijama (npr. renderovana, koja ima link na sliku u originalnoj dimenziji, prikaz preko celog ekrana, sortiranje slika po nekom parametru, iščezavanje i sl.). Da bi se adaptirali, video i audio materijali treba da budu fragmentirani, opisani i semantički povezani. Na taj način, fragmentima se može manipulirati u smislu dodavanja, skrivanja, izmene, sortiranja (kao sa tekstom). Potpuna adaptivnost multimedijalnih sadržaja je veoma kompleksna, zavisi od formata fajlova, verzije kodeka za čitanje sadržaja, mogućnosti hardverske platforme (audio i video komponenti/kartica) i mogućnosti aplikacija na klijentskoj i serverskoj strani (verzije, kompatibilnost, itd).

3.4.3.2 Adaptacija navigacije

Adaptivno vođenje kroz sadržaj ima šest pod-tipova. Direktno vođenje (*direct guidance*) isključuje slobodnu navigaciju studenta u prostoru znanja. Adaptivno sortiranje linkova (*adaptive link sorting*) predstavlja drugi kraj skale; navigacija je potpuno slobodna, a sortiranje olakšava snalaženje studenata u prostoru znanja. Adaptivno skrivanje linkova (*adaptive link hiding*) ograničava navigaciju studenta skrivanjem, prikazivanjem, onesposobljavanjem, ili uklanjanjem linkova u zavisnosti od statusa (rezultata) studenta.



Ilustracija 39: Taksonomija navigacione adaptacije

Adaptivno anotiranje linkova (*adaptive link annotation*) predstavlja označavanje statusa linkova koji su predstavljani studentu. Na primer, sadržaji (linkovi) koji su naučeni/učeni/posećeni/neposećeni označeni su različitim bojama, a stepeni naučenosti su grafički/tekstualno označeni procentima, ili menzuirom. Adaptivno generisanje linkova (*adaptive link generation*) slično je adaptivnom skrivanju linkova, s tim da se generišu linkovi koji prethodno nisu kreirani, a skrivanje to čini sa postojećim linkovima. Adaptacija mapa (*map adaptation*) je najkompleksniji oblik navigacione adaptacije. Hipermedijalni prostor znanja u AHE sistemima, može se razmatrati kao usmereni graf - mapa u kojoj sadržaji učenja predstavljaju čvorove, a linkovi za navigaciju čine lukove, koji povezuju te čvorove. Ako postoji potreba da se ova mapa prikaže studentu, onda se ona prethodno adaptira prema definisanoj pedagogiji i stanju studentskog modela, a zatim se isporučuje studentu. Na taj način, zadržava se utisak da je navigacija potpuno slobodna, pošto nema vidljivih ograničenja za pretragu prezentovane mape. Promena se primećuje

tek u novoj sesiji korisnika sa sistemom. Studentski model se ažurira pri svakom logovanju, a to proizvodi adaptaciju mapa sa sadržajima učenja.

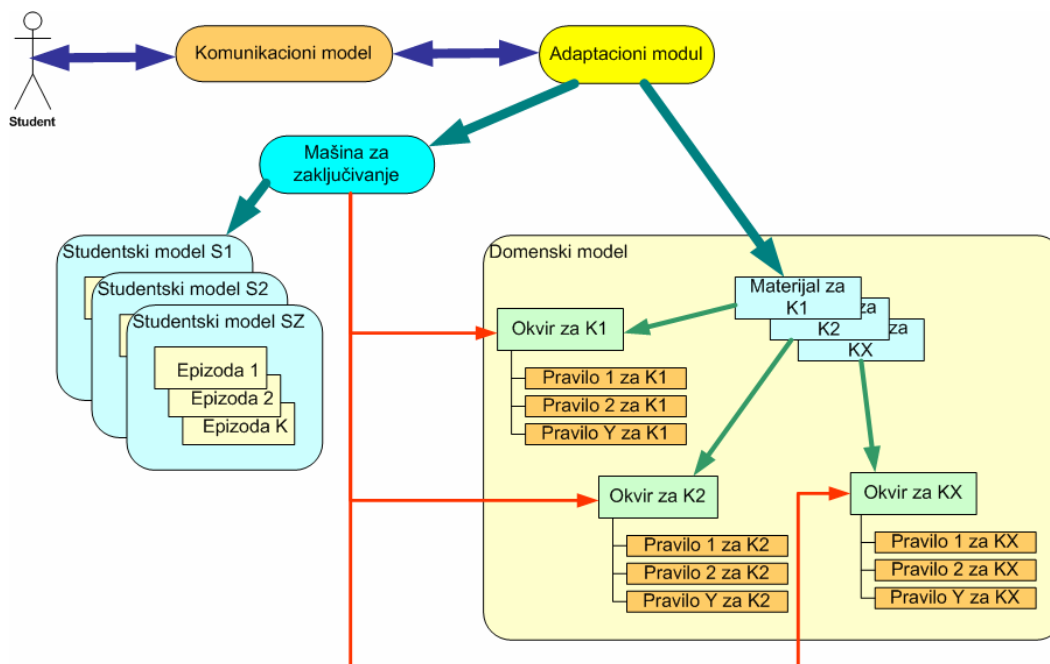
3.4.4 Primeri adaptivnih edukativnih hipermedijalnih sistema

3.4.4.1 ELMART

ELMART (Episodic Learner Model the Adaptive Remote Tutor) neki autori svrstavaju u ITS sisteme. Obzirom da je fokusiran na izučavanje nastavnih materijala, koji sadrže znanje predstavljeno u deklarativnoj formi, u novijim publikacijama se ovaj sistem svrstava u adaptivni edukativni hipermedijalni sistem. *ELMART* je zasnovan na Web tehnologijama, namenjen za podučavanje studenata u *LISP* programiranju [55, 64, 65]. On obezbeđuje referentne materijale, primere i probleme predstavljene u hipermedijalnoj formi, omogućujući studentima da pretražuju bilo koji deo hipertekst prostora.

Adaptivnost *ELMART*-a ogleda se u adaptivnoj podršci za navigaciju (adaptivna anotacija linkova), kao i adaptivnom programu (redosledu) izvođenja nastave (direktno vođenje) [66]. Drugom rečju, od samog početka nisu dostupni ni svi nastavni materijali, ni svi testovi. Student postepeno ovladava znanjima o *LISP* programiranju. Nakon logovanja u sistem, nakon završetka jedne celine (lekcije i/ili vežbanja), student dobija listu naslova koje dalje može proučavati. Ta lista je adaptibilna.

Da bi sistem bio adaptivan, domensko znanje predstavljeno je hijerarhijskom (stablastom) strukturom. Svaki čvor stabla sadrži podatke o preduslovima koje student treba da ispuni da bi mogao da poseti čvor, koncepte znanja vezane za čvor i koncepte (adrese čvorova) za koje se pretpostavlja da mogu biti posećeni nakon posete tekućeg čvora.



Ilustracija 40: Arhitektura ELMART sistema

Arhitektura *ELM ART* sistema je specifična (Ilustracija 41). Ta specifičnost se ogleda u kompleksnom domenskom modelu i modelu studenta. Domenski model je fragmentiran na domenske koncepte, koji predstavljaju najmanje i nedeljive celine znanja. Domenski koncepti se sastoje od dve celine:

- Sadržaji (materijali) za učenje – deklarativno znanje iz *LISP* programiranja predstavljeno je kroz definicije, objašnjenja i primere korišćenja (na slici označeni kao *materijal za K.*).
- Konceptni okviri (*Frame-ovi*) – objekti koji sadrže kroz schema-e formalizovano znanje o sintaksi jezika, algoritmima i o načinu (šabloni) rešavanja problema. Pored toga, okvirima su pridružena pravila kojima su opisani različiti načini rešavanja problema vezanih za konkretan koncept. Postoje dve vrste pravila: pravila korektnog toka (*solution rules*) i pravila za greške (*buggy rules*).

Epizodinčni model *ELM* (*Epizodic Learner Model*) je korišćen kao studentski model. Suštinu modela čine *epizode* – delovi sesija korisnika sa sistemom. Za svaki problem, ili zadatak, koji student pokuša da reši, pokušaj sa greškama i korektnim akcijama se pamti u epizodički model. Zaključivanje se vrši u mašini za zaključivanje nad sistemom pravila (u kojima se vrši transformacija domenskog znanja u oblik komparativan epizodičkom modelu), i upoređivanjem epizodičkog i prevedenog domenskog modela. Zaključke koristi modul za adaptaciju, koji zatim bira odgovarajuće materijale, koje prosleđuje studentu posredstvom komunikacionog modela.

Za svakog novog studenta sistem kreira inicijalni studentski model na osnovu upitnika (Ilustracija 42a) koga student popunjava nakon prvog logovanja u sistem, i inicijalnog testiranja (Ilustracija 42b) u skladu sa popunjenim upitnikom. Upitnik je sastavljen od tri pitanja koja treba da sagledaju minimalno predznanje studenta i iskustvo u korišćenju računara i Web pretraživača. Time se student svrstava u neki od stereotipova, na osnovu koga sistem preduzima odgovarajuće akcije.

Intro-Questionnaire

ELM-ART is a new, intelligent system that allows for interactive learning via WWW. The development of the system is just in an experimental phase investigating different ways of knowledge-based support. Therefore, data gathered during working with this system are evaluated statistically (for scientific purposes only).

To interpret data correctly, we ask you for answering the following questions. In return you get the possibility to work at all six lessons of the introductory LISP course. If you don't answer the questions, you will be able to play with the first lesson only. However, at any moment within the course, you can go to the intro page (the system's home page, that is the next page where you go to from here) and ask for the other lessons in the preferences section.

Experience in

working with WWW browsers:	programming languages:	using computers:
☐ none	☐ none	☐ never before
☐ little	☐ LISP	☐ up to 20 hours
☐ something	☒ Pascal, C, C++, Basic	☐ 20-100 hours
☐ much	☐ others	☐ more than 100 hours

a) Inicijalni upitnik za novog korisnika

Pre-Test
to "Datatypes"

Please, work at the following items:

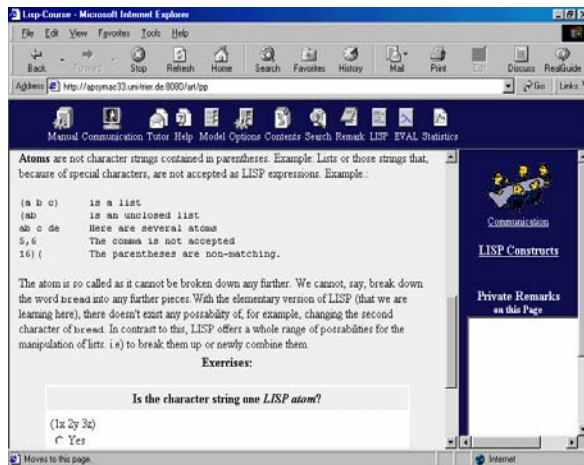
Is the character string one LISP atom?
453
☐ Yes
☐ No

Is the character string one symbolic atom?
jack-jenny
☐ Yes
☐ No

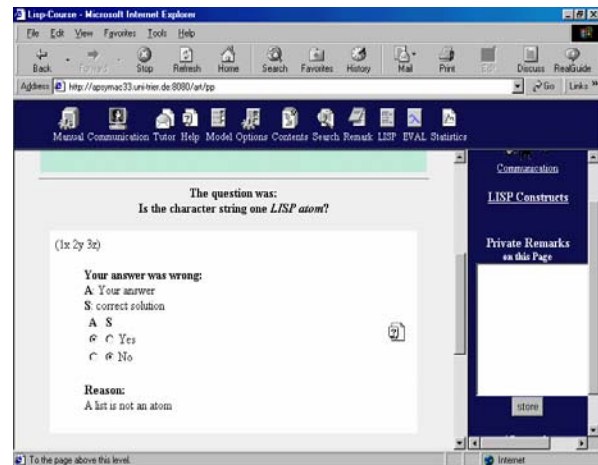
Is the character string a number?
123
☐ Yes
☐ No

Show up pages that may help to solve the exercise item.

b) Pre-Test



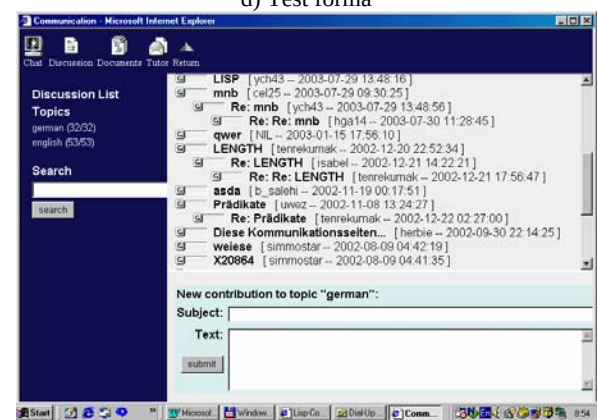
c) Nastavni materijal



d) Test forma



e) Obrazložjenje rezultata



f) Diskusiona grupa

Ilustracija 41: ELMART - izgled interfejsa

Na kraju svakog skupa vežbi, student dobija detaljne informacije o učinjenim greškama (Ilustracija 42c,d,e). Podaci o greškama sistem koristi za potrebe korigovanja znanja studenta. Istorija studentskih performansi određuje specifičnost prikaza povratne poruke. Studentski model održava model učenja ažurnim zahvaljujući čuvanju rezultata za svaki posećeni čvor. Student može višestruko posećivati isti čvor tako da njegov uspeh (na čvoru) fluktuira u zavisnosti od korisničkog unosa. Ti podaci se pohranjuju u glavnom fajlu i mere se kako bi odrazili različite aspekte studentskog znanja. Sistem pruža mnogobrojne pomoćne servise studentu, kao što su:

- Korišćenje servisa diskusionih grupa (podrška kolaborativnom učenju - Ilustracija 42f)
- Uvid u koncepte ontologije,
- Statistika napredovanja studenta,
- Uvid u lični model kao i njegovu delimičnu reviziju (u slučaju pogrešno popunjenog polaznog upitnika - mislio sam da sam mnogo bolji nego što stvarno jesam),
- Mogućnost vođenja beleški za svaku lekciju (*Private Remarks*),

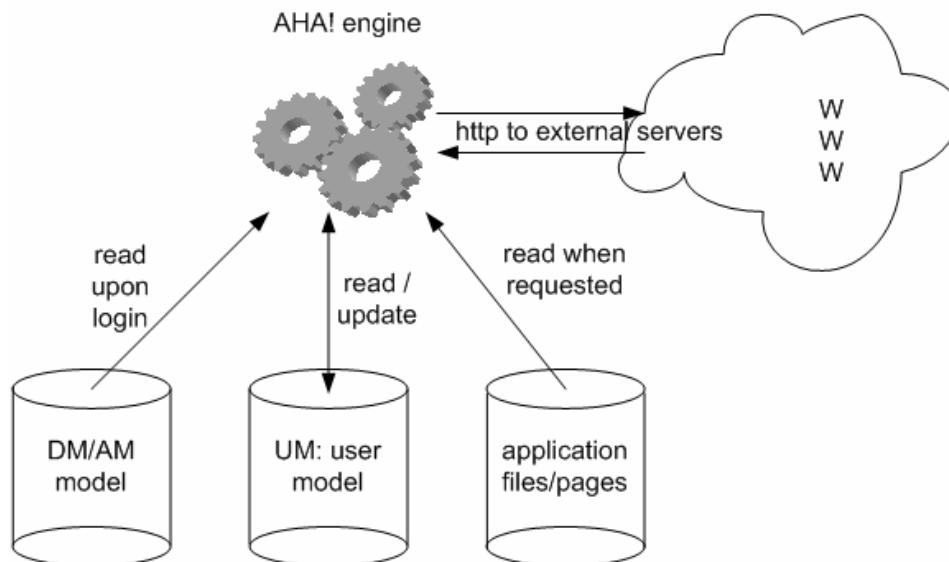
- Mašinu za pretraživanje nastavnih materijala i vlastitih beleški po specificiranim kriterijumima (ključne reči ili izrazi).

ELM ART je aktuelan *AHE* sistem, koji prati zahteve vremena ne napuštajući početne koncepte i idejnu osnovu sistema. U vreme nastajanja prve verzije, klasifikovan je kao *ITS* obzirom da ima vrlo robustan studentski model i mogućnost učenja rešavanjem problema. Fokus ovog sistema je, međutim, na nastavnim materijalima. Glavni cilj testova i rešavanja zadataka je dobijanje što potpunije informacije o studentu, radi adaptacije sadržaja za učenje.

3.4.4.2 AHA!

AHA! sistem omogućava adaptaciju sadržaja, stila prikaza i anotaciju linkova. Za razliku od većine drugih *AEH* sistema, *AHA!* je domenski nezavisan sistem. Jezgro sistema predstavlja komponenta imenovana kao *AHA! engine* (Ilustracija 43). Ova komponenta se sastoji iz dva dela:

- Mašina za zaključivanje na bazi pravila (*reasoning/rule engine*) – komponenta koja ažurira model studenata prevođenjem, na bazi akcija studenta.
- Adaptivni selektor resursa (*adaptive resource selection*) – komponenta u kojoj je implementirano uslovno određivanje linkova kao navigaciona tehnika, i uslovno uključivanje objekata kao prezentaciona tehnika.



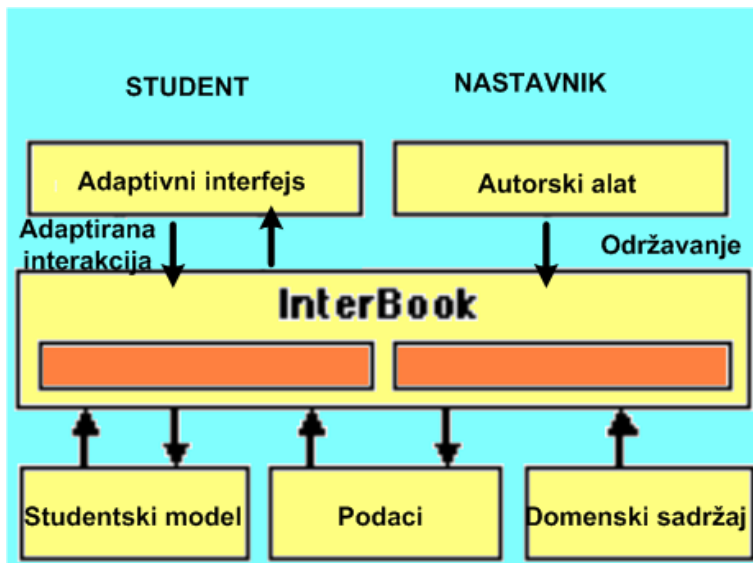
Ilustracija 42: *AHA!* sistem – generalna arhitektura

Pored navedenih komponenti, postoje:

- *DM/AM model* – konceptualna struktura sistema implementirana je u domenske modele i modele adaptacije; ovim modelom su obuhvaćena i adaptaciona pravila
- *UM* – studentski model, koji se ažurira svaki puta kada student načini neku akciju; ove akcije su u *AEH* sistemima uglavnom zahtev za sadržajima učenja, pri čemu se potražuju fajlovi iz lokalnih repozitorijuma, ali mogu da se referenciraju i fajlovi iz spoljnjih izvora.

3.4.4.3 InterBook

AEH sistem *InterBook* je jedan od najviše referenciranih materijala u literaturi o adaptivnim hipermedijalnim sistemima [124]. U *InterBook*-u se primenjuju anotacija linkova i sortiranje kao tehnike adaptacije. Ovaj domenski nezavisni sistem je karakterističan jer je autorski alat (Ilustracija 44) sastavni deo sistema, a ne nezavisno dizajnirana aplikacija (slučaj kod većine ITS i AEH sistema). Osnova adaptivnih sistema, pa tako i *InterBook*-a, je studentski model koji pohranjuje aktuelno stanje znanja studenta. U *InterBook*-u je korišćen prošireni prekrivajući studentski model (poglavlje 2.2.2). Ovaj model se formira na osnovu ciljnog domena (znanje se unosi posredstvom autorskog alata).



Ilustracija 43: Arhitektura AEH sistema *InterBook*

Čitanje materijala, pregledanje primera, i rešavanje pitanja na bazi ponuđenih odgovora su osnovni izvori informacija o studentu. Ovi podaci se pamte u bazu podataka. Njihova višedimenzionalnost čini proširenje osnovnog prekrivajućeg modela studenta (osnovni model koristi samo jednu dimenziju – rezultate testova).

$$K(c) = \frac{\sum_s (w_s * v_s(c) - t_s)}{N_s} \quad (1)$$

Skalarna vrednost znanja studenta $K(c)$ za konkretan koncept obuhvata sve dimenzije (kriterijume) koji se beleže u studentskim logovima, i izračunava se (Izraz 1) kao količnik sume *izvedene vrednosti* i broja dostupnih rezultata (N_s). Za svaku dimenziju računa se proizvod težinskog faktora kriterijuma (w_s) i rezultata (za konkretan koncept) $v_s(c)$. Taj proizvod se dalje umanjuje za vrednost praga – minimalnog rezultata koji se razmatra kao prolazni (t_s). Ova kalkulacija se obavlja svaki put kada korisnik završi neku proveru, kako bi se ponovo izvršila adaptacija.

InterBook je jedan od prvih AEH sistema. Svojim originalnim (onovremenskim) rešenjima, predstavljao je jednu od najvažnijih implementacija, posredstvom kojih su prikupljana iskustva za nove sisteme, istraživanja i standarde. Ograničenost *InterBook*-a

je preciznost studentskog modela da odredi realno stanje studenta. Koristi se samo jedan tip testiranja, pogodan za proveravanje deklarativnog znanja. Za egzaktno nauke potrebno je proširenje za učenje rešavanjem problema. Iz tog razloga realizovano je prvo povezivanje nezavisnih sistema za učenje na daljinu – povezivanje *InterBook*-a i *Cognitive Tutor*-a (preciznije u prethodnoj verziji ovog sistema pod nazivom *PAT OnLine*) (Poglavlje 3.2.2). Na taj način sistemi su se međusobno upotpunili – student stiče deklarativno znanje u *InterBook*-u, a praktično znanje iz istog domena stiče rešavanjem zadataka u *PAT OnLine*. Interoperabilnost ova dva sistema ogledala se u razmeni podataka između studentskih modela i sinhronizovanim adaptivnim vođenjem (pošto se stekne deklarativno znanje, prelazi se na učenje rešavanjem problema).

3.5 Sistemi za upravljanje učenjem

Sistemi za upravljanje učenjem (u daljem tekstu *LMS* – *Learning Management Systems*), je, po broju implementacija i masovnosti korišćenja, najveća grupa sistema za e-učenje. Ovi sistemi nisu adaptivni, ali zbog velike palete administrativnih funkcionalnosti, imaju veliku primenu u formalnom obrazovanju (u edukativnim institucijama). *LMS* obezbeđuju kompletnu platformu za održavanje korisničkih naloga, organizovanje korisnika prema ulogama u sistemu i na osnovu obrazovnih profila, za kontrolu pristupa resursima učenja. Pored toga ovi sistemi nude alate za kreiranje, publikovanje i definisanje načina korišćenja nastavnih sadržaja. *LMS* omogućavaju konstruktivistički pristup učenju, kroz paletu kolaborativnih alata za sinhronu i asinhronu komunikaciju između korisnika [56]. *LMS* imaju tri osnovne grupe funkcionalnosti:

- Nastavničke funkcionalnosti – kreiranje, opisivanje i objavljivanje resursa učenja, organizovanje resursa u predmete, nastavne jedinice (teme, lekcije), provere, saradnja sa drugim nastavnicima, studentima, praćenje napredovanja (rezultati i performanse) studenata.
- Studentske funkcionalnosti – korišćenje resursa učenja (proučavanje publikovanih materijala, proveravanje), saradnja sa drugim studentima i nastavnicima (različiti oblici sinhrono i asinhrono komunikacije).
- Administratorske funkcionalnosti – upravljanje celokupnim sistemom (upravljanje nalogima korisnika, održavanje baze podataka i fajl sistema).

Zbog velikog broja funkcionalnost, arhitektura *LMS* je mnogo kompleksnija nego kod adaptivnim sistemima (Ilustracija 45). *LMS* su najčešće modularni. Jezgro sistema čini grupa modula za osnovne funkcionalnosti, koji su obavezni. Ostatak sistema čine moduli za dodatne funkcionalnosti, koji mogu po potrebi da se uključuju, ili isključuju. Osnovni moduli sistema su:

- Modul za upravljanje sadržajem (*Content Management Module*) – preko ovog modula se upravlja sadržajima na lokalnom repozitorijumu. Najčešće se radi o fizičkom smeštanju fajlova na fajl sistem, dok se podaci koji opisuju resurse (fajlove) čuvaju u BP. Radi očuvanja konzistentnosti i integriteta, za resurse koji se nalaze van sistema, prave se kopije u sistemu. Na taj način, *LMS* ne zavisi od upravljanja resursima van sistema.
- Modul za raspoređivanje (*Sequencing Module*) – ovaj modul namenjen je za definisanje veza između sadržaja učenja. Takođe, preko ovog modela se povezuju

sadržaji učenja sa testovima. Definisanjem relacija, praktično se definiše redosled pojavljivanja resursa pri korišćenju sistema. U najvećem broju praktičnih primena, koriste se standardni formati (vidi poglavlje 3.2.3, *IMS SS*).

- Modul za testiranje (*Assessment Module*) – pored definisanja provera, ovaj modul omogućava definisanje načina eksploatacije testova u različitim fazama nastavnog procesa. Modul za testiranje pruža i servise modulu za praćenje studenata. Iz tog aspekta, on predstavlja izvor podataka o studentu (rezultata provera).
- Modul za administriranje kurseva (*Course Management Module*) – ovaj modul je glavni modul za organizovanje predmetne nastave (kurseva). Preko ovog modula određuje se struktura predmeta – definišu se nastavne jedinice (teme, lekcije), provere, a zatim se ovi entiteti povezuju sa odgovarajućim resursima.
- Komunikacioni model (*Communication Model*) – ovaj deo sistema je odgovoran za pouzdanu distribuciju resursa učenja svim korisnicima. Svi aktuelni *LMS*-ovi su Web aplikacije, tako da je komunikacioni model tesno povezan sa tehnologijama koje se koriste za generisanje Web sadržaja (Web stranica). Iako je ovaj modul, iz ugla programera i dizajnera sistema, marginalan i lako zamenjiv, pokazalo se da ima vrlo važnu ulogu u oceni sistema od strane korisnika (što je potpuno razumljivo jer je to za najveći broj korisnika jedino vidljiv deo sistema).
- Modul za praćenje (*Tracking Module*) – ovaj modul ažurira studentske zapise akcijama studenta pri korišćenju sistema. Regstruju se statistički podaci kao što su broj pristupanja konkretnim sadržajima učenja, vreme provedeno na pojedinim stranicama, navigacione putanje, regularnost akcija u fazama testiranja, rezultati provera. Ovaj modul najčešće manipuliše neobrađenim podacima.
- Studentski model – kao kod adaptivnih sistema, studentski model namenjen je čuvanju podataka svakog individualnog studenta (*elektronski studentski dosije*). Prethodno opisan modul za praćenje je odgovoran za hranjenje studentskog modela podacima. Do sada, u *LMS* sistemima, malo pažnje se posvećivalo dizajnu studentskog modela. Pojavom međunarodnih standarda za modelovanje korisnika u nastavi (poglavlje 3.1), i potrebama edukativnih institucija da poseduje konzistentne podatke studenata, koji se mogu razmenjivati sa drugim sistemima, doveo je do izdvajanja studentskog modela kao posebnog dela *LMS*-a.

- Moduli aktivnosti – modul za dodelu obaveza, modul radionica, modul za kvizove.
- Moduli za kolaboraciju – ovde spadaju moduli za *chat*-ovanje, forume i ankete.

Zahvaljujući implementaciji koja je realizovana u *PHP* skript jeziku (koji je masovno prihvaćen u Interent zajednici), postoji veliki broj dodatnih modula, kreiranih od strane *open source* programera (na primer moduli rečnika, žurnala, dijaloga i lekcija).



Ilustracija 45: Izgled početne stranice kursa za pravljenje igara kreiranog u Moodle-u

U Moodle-u je implementiran socijalni konstruktivizam, kao pedagoški pristup edukaciji (Poglavlje 2.1.1). Ovaj pristup potencira socijalni aspekt učenja, tako da je kolaborativnim aspektima (moduli za kolaboraciju) dat podjednak značaj u realizaciji sistema, kao aspektima organizacije nastave i upravljanju resursima učenja. Organizacija nastave u Moodle-u je zasnovana na *SCORM 2004* specifikaciji. Nastava se odvija kroz aktivnosti, za koje se definiše kalendar, dodeljuju obaveze sa trajanjem i rokovima. U realizaciji obaveza, studenti se organizuju u grupe, tako da mogu međusobno da sarađuju u kontekstu obaveza, a rezultati se evaluiraju i na grupnom i na pojedinačnom nivou. Nastavni materijali su strukturirani po *IMS CP* specifikaciji, dok se kvizovi pamte u *IMS QTI* formatu.

Osnovne karakteristike Moodle-a su:

- Lako kreiranje kurseva korišćenjem postojećih resursa,
- Kreiranje sadržaja kurseva koje mogu višestruko koristiti različiti korisnici, uključujući i sadržaj ostalih sličnih sistema (kao što su WebCT ili Blackboard),

- Zahteva dodatno angažovanje učenika i samostalan rad,
- Obezbeđuje prijavljivanje i autentifikaciju korisnika na bezbedan način,
- Posедуje aktivan sistem podrške za pomoć u rešavanju problema i kreiranju novih ideja,
- Lako dostupan, s obzirom na trenutni razvoj internet tehnologija.

Moodle je modularno dizajniran što ga čini veoma pogodnim za kreiranje novih kurseva i dodavanje novog sadržaja, koji će dodatno angažovati učenike[5]. Moodle se karakteriše veoma interaktivnim interfejsom koji instruktorima omogućava lako kreiranje kurseva (slika 12). Kreiranje kurseva je jednostavna operacija i obuhvata kreiranje privilegija i odgovornosti određenog korisnika nad kreiranim kursom, selektovanje jednog od tri moguća formata kurseva: tematski (engl. topics), socijalni (engl. social), ili nedeljni (engl. weekly), i na kraju se, u već postojeći šablon, ugrađuju sadržaji kursa.

3.5.2 WebCT i Blackboard

WebCT je prvi uspešan komercijalni proizvod za e-učenje. Implementiran je kao Web aplikacija – kolekcija *Perl* skriptova. Sadržaj i konfiguracioni podaci su, u prvim verzijama bili smešteni u tekstualnim fajlovima, bez *XML* podrške, ali je upravljanje resursima bilo zadovoljavajuće. Modularna arhitektura je podelila funkcionalnosti po različitim, međusobno nezavisnim modulima (alatima). *WebCT* je imao vrlodobre komunikacione (kolaborativne) karakteristike. Kursevi su se organizovali u pakete – takozvani *e-packs*. Najveća slabost sistema je bila komplikovan korisnički interfejs. Kreatori su nastojali popraviti stanje dodavanjem pomoćnih alata (čarobnjaka – vizards), kako bi pojednostavili produkciju kurseva. Od svog nastanka (1996), sistem je tokom istorije kontinualno rastao (npr. 10 miliona studenata u 80 zemalja, 2002) i menjao vlasnike. Najnovija verzija ovog *LMS* (*WebCT Vista*) predstavlja integraciju *WebCT* –u *Blackboard LMS* sistem.

Blackboard [126] je drugi primer komercijalnog *LMS*. Dizajniran je kao skup alata za podršku različitim funkcionalnostima učenja. Ovi alati se mogu tretirati slabo spregnutim sistemskim modulima. Najvažniji među njima su:

- *My Faculty Tools* – modul namenjen nastavnicima za dizajn kurseva. Ponuđene su razne funkcionalnosti: postavljanje i objavljivanje postojećih resursa, kreiranje novih resursa, komponovanje i deskripcija sadržaja metapodacima. Kurs se interno predstavlja paketom zvanim *Course Cartridge*. Slično kao kod *WebCT*, ovaj paket se komponuje od pojedinačnih resursa koji su opisani i upakovani sa opisom u kompaktne celine zvane *building blocks*.
- *BuildingBlock Catalog* – modul koji omogućava pretragu resursa učenja koji su predstavljeni *building blocks*-ovima. Posredstvom ovog alata omogućen je pristup repozitorijumu resursa, kao i mogućnost višestrukost korišćenja resursa u sistemu.
- *Search Content Tool* – alat koji omogućava studentima pretraživanje resursa kurseva. Ovaj alat takođe omogućava pristup repozitorijumu resursa, ali u određenom nastavnom kontekstu.

- *Seneca Toolkit* – skup alata koji su dizajnirani za administriranje korisnika sistema. Ima veliku paletu funkcionalnosti, od kojih su najvažnije upravljanje grupama korisnika, pojedinačnim korisničkim nalogima, definisanje privilegija za korišćenje različitih sistemskih resursa.
- *LinkMaker* – predstavlja alat koji omogućava povezivanje resursa učena i nastavnih jedinica – sadržaja i organizacije.
- *FAQTool* – predstavlja alat za upravljanje listama čestih pitanja i odgovora (*Frequently Asked Questions – FAQ lists*). U *Blackboard*-u, liste su grupisane u skladu sa sadržajem koji je dostupan studentima (publikovan i povezan sa kursevima). Ovaj alat je rezultat rešavanja praktične potrebe, da se delimično automatizuje komunikacija nastavnika i studenata, i da se pruži efektivnija pomoć u korišćenju sistema.

Elektronska pošta i forumi predstavljaju kolaborativnu komponentu sistema. *Blackboard* je *LMS* koji je težišno orjentisan na sadržaje učenja. Veći deo modula je namenjen kreiranju, organizovanju i distribuciji resursa. Socijalni aspekti, kao što su međusobna komunikacija, grupne obaveze, ili *chat*-ovanje nisu obuhvaćene u ovo sistemu. Obzirom na takve karakteristike, uočena je kompatibilnosti sa *WebCT* sistemom, tako da je 2006.godine izvršena fuzija ova dva sistema. Pored toga, razvijen je poseban sistem za socijalnu komunikaciju *Blackboard CommunitySystem*. Ovaj, potpuno tržišno orjentisan sistem (890 zaposlenih do juna 2008, ukupni prihodi preko 19 milona dolara), pored sistema za e-učenje, sadrži i druge proizvode: transakcioni sistem za univerzitete, sistem e-trgovine i transakcioni sistem za debitne kartice.

Na osnovu prethodnih razmatranja, može se zaključiti o dve vrste ovih sistema – komercijalni sistemi, i besplatni – takozvani *open source* sistemi. Zainteresovani za korišćenje *LMS* se najčešće nalaze u dilemi prilikom nabavke – za koju od ove dve opcije da se opredele? Postoje prednosti i nedostaci u oba slučaja. Prednosti komercijalnih sistema su u postojanju garancija za njihovu pouzdanost, održavanje, i predstavljen opis funkcionalnosti. Pored softverskog rešenja, proizvođači nude sistem za upravljanje BP i hardversku opremu, a u prodaju je uključena i obuka korisnika. Nedostatak istih je u (za nas) visokoj ceni koštanja, koja se izražava paušalno, ili zavisi od broja korisnika sistema (broja studenata). Proširljivost ovih sistema predstavlja dodatne finansijske troškove, kao i zamena postojećih verzija softvera novijim.

Kod *open source* sistema ne postoji finansijski motiv proizvođača, ali se naplaćuju usluge za podešavanja sistema konkretnim zahtevima. Proizvođači se licencom oslobađaju odgovornosti od šteta (nefunkcionisanja, gubljenja podataka i sl.) koje mogu da se dogode. Sa druge strane, logika sistema je potpuno transparentna, tako da se mogu implementirati nove i prilagođavati postojeće funkcionalnosti sistema. Uvek postoji potencijalna ranjivost zbog otvorenog i dostupnog izvornog koda. Ažuriranje novim verzijama sistema nije problematično jer u skoro svim slučajevima postoji kompatibilnost.

Pored datih primera *LMS*, postoje i mnogi drugi (npr. *ATutor* [127], *Olat* [128], *Sakai project* [129], *Dokeos* [130], *Desire2Learn* [131]). Skoro svi savremeni *LMS* implementiraju međunarodne standarde u oblasti e-učenja, naročito za opisivanje i pakovanje nastavnih materijala (sadržaja učenja). S druge strane, mali je broj sistema

koje se njih striktno pridržavaju. Kod većine je kreiran vlastiti IEEE LOM ili IMS LRM aplikacioni profil, odnosno postoji delimično oslanjanje na postojeće standarde. Uvode se sopstveni elementi i kategorije, što ima za posledicu niži stepen interoperabilnosti heterogenih aplikacija. Osim toga, tipična situacija je da različite organizacije koriste termine iz različitih vokabulara pri dodeli vrednosti elementima propisanim standardom, što dodatno ugrožava interoperabilnost.

Drugo zajedničko svojstvo je da podržavaju različite koncepte učenja i korisnicima nude veoma slične palete funkcionalnosti. Iz aspekta dizajna i implementacije, takođe postoje sličnosti – većina ima modularnu strukturu, sa zajedničkom BP. Većina *LMS* je implementirana u *PHP* jeziku, tako da ne postoje jasne granice između interfejsa, logike sistema i vezama prema BP. Za korisnike *open source* sistema, koji sami moraju da prilagođavaju sistem svojim potrebama, ovo je izuzetno otežavajuća okolnost. Implementacija proširenja dodavanjem novog modula izgleda suštinski jednostavna. Međutim, zbog kontrole sesije, provere statusa i stanja korisnika (studenta ili nastavnika) u toku eksploatacije, bezbedne razmene informacija između korisnika i sistema, kao i modula unutar sistema, potrebno je povezati novi sa postojećim modulima. U praksi se pokazalo da ovaj posao nije nimalo lak.

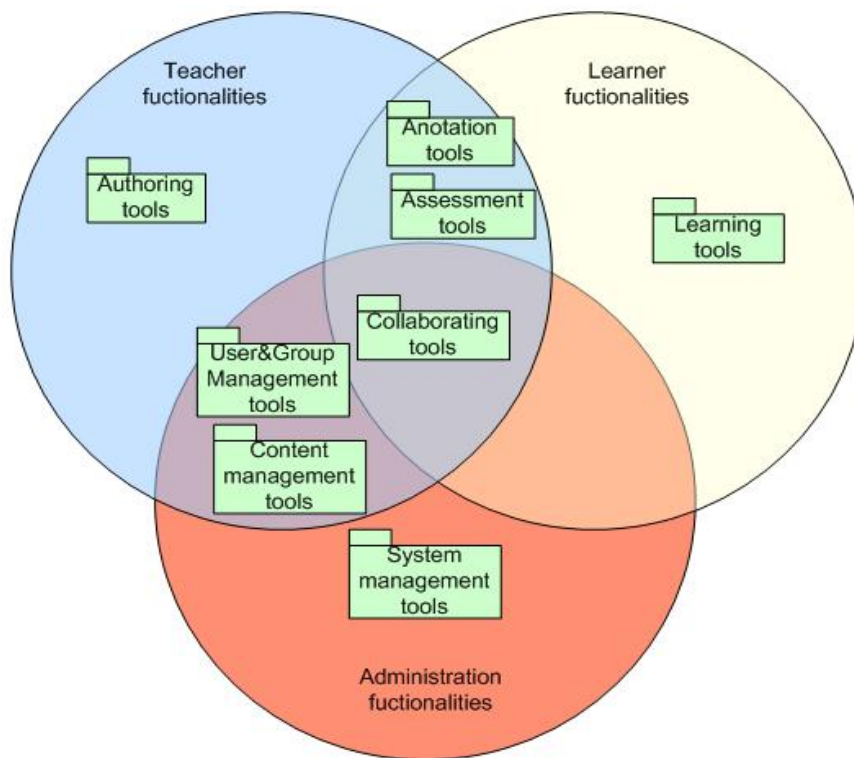
4 Prikaz predloženog rešenja

Veliki broj funkcionalnosti, primena standarda, prezentaciona i navigaciona adaptivnost, bezbednost sistema od zlonamernih korisnika i upada, predstavljaju najznačajnije od velikog broja karakteristika, koje se nameću kao nužne kod savremenih sistema za e-učenje. Da se ne bi kompleksnost karakteristika prenela na kompleksnost dizajna i implementacije, model zasnovan na međusobno slabo spregnutim komponentama predstavlja optimalno rešenje ovako složenog sistema. Pored navedenog, razmatrane su još dve alternative: kompaktan sistem i sistem baziran na nezavisnim alatima.

Kompaktna arhitektura sistema omogućava lakšu prenosivost sistema na različite računarske platforme (u pogledu hardvera i sistemskog softvera). Oni najčešće funkcionišu kao samostalne (*stand alone*) aplikacije koje imaju sopstvene podatke i sadržaje, koji nisu deljivi, mada se mogu izvoziti i uvoziti u nekom internom, ili standardnom formatu (poglavlje 3.2). Modifikacije i proširenja su gotovo nemoguća, već se za ovo svojstvo koristi tzv. verziranje – postavljanje nove verzije sistema, koje prate nove verzije neophodnog sistemskog softvera. Primeri ovog dizajna su česti kod sistema za upravljanje učenjem (poglavlje 3.5.2) i kod većine adaptivnih sistema za e-učenje (poglavlje 3.4.2.2 i 3.4.4.1). Dobra karakteristika arhitekturno kompaktnih sistema je i da imaju dobre preformanse za implementirane funkcionalnosti. Nemogućnosti ugnježdavanja u sistem domaćina i korišćenje zajedničkih, deljivih podataka predstavlja slabost ovakve arhitekture.

Arhitektura zasnovana na nezavisnim alatima predstavlja drugi kraj skale softverskih rešenja kod sistema za e-učenje. Pored navedenih sistema za e-učenje (poglavlja 3.4 i 3.5), postoji paleta samostalnih aplikacija koje su specijalizovane za samo neke od funkcionalnosti e-učenja [78]. Primer takvih alata su aplikacije za stuktuiranje kurseva u SCORM standardu: Reload editor [79], eXe-learning [80], Docyrus [81]. Drugi primer samostalnih alata su aplikacije za pravljenje testova: QuizCreator [82], Quiz Builder [83], Hot Potatoes [84]. Pored navedenih postoje i aplikacije specijalno dizajnirane za određeni

LMS (poglavlje 3.5), ITS, ili AHES (poglavlje 3.4). Na primer, CTAT [85] je skup autorskih alata za ITS Cognitive tutor (poglavlje 3.4.2.1). ToolBook [86] predstavlja skup alata za Docent LMS. Postoje i autorski alati opšte namene koji podržavaju standardne formate (npr. IMS – poglavlje 3.2.3, ili SCORM – poglavlje 3.2.4), tako da nisu posebno vezani za pojedinačne sisteme za e-učenje. Primeri takvih alata su Adobe Authorware [87] i Lectora [88] za sisteme po SCORM standardu. Svi nezavisni alati mogu da budu kategorisani u tri glavne grupe (Ilustracija 47) na osnovu njihovih funkcionalnosti: nastavnički alati, studentski alati i administratorski alati.



Ilustracija 46: Klasifikacija alata za elektronsko učenje

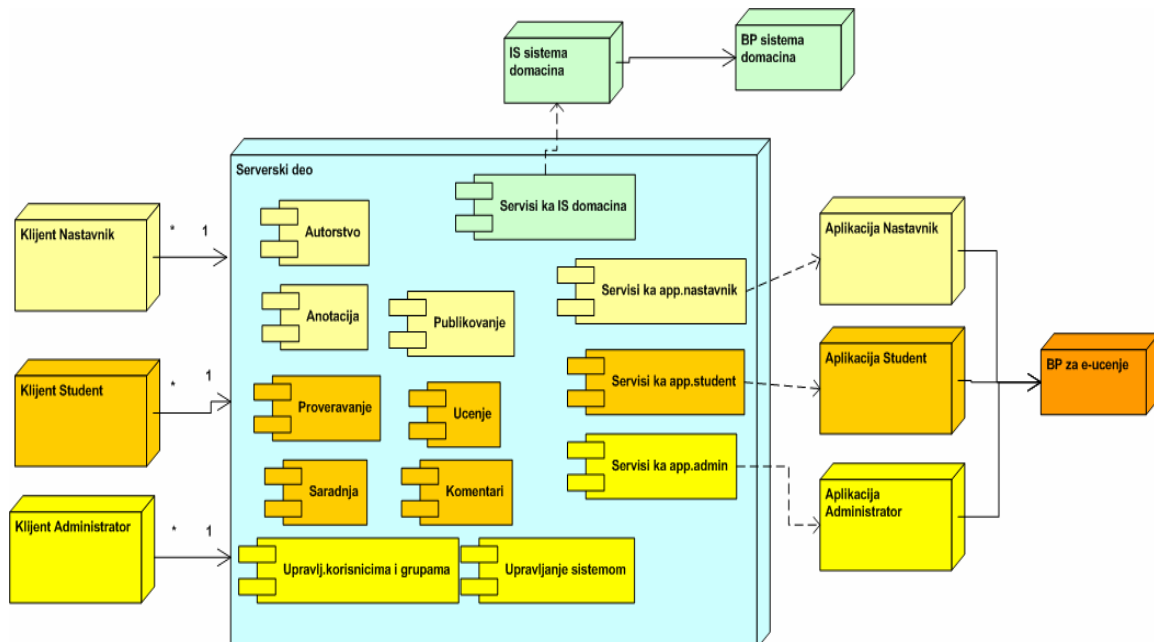
Alati za saradnju (*Collaborating Tools*) su namenjeni za uspostavljanje različitih vrsta komunikacija i saradnje među korisnicima (na primer, e-pošta, forumi, oglasne table, obaveštenja, diskusione grupe, itd.). Anotacioni alati (*Annotation Tools*) omogućavaju nastavniku da naznače važne delove materijala za učenje, da daj dodatna objašnjenja za iste. Studentski alati za anotaciju omogućavaju studentima da unose beleške i na taj način da personalizuju sadržaje učenja. Korišćenjem alata za testiranje (*Assessment Tools*) i za upravljanje korisnicima (*User&Group Management Tools*), nastavnici mogu da analiziraju napredovanje studenata. Administratori upravljaju kompletnim sistemom za e-učenje korišćenjem alata za sistemski menadžment – *System Management Tools* (na primer, upravljanje korisničkim nalogima, podešavanjem i konfigurisanjem sistema, administriranje BP i sl.). Administratori i nastavnici upravljaju različitim vrstama sadržaja korišćenjem alata za upravljanje sadržajem (*Content Management Tools*). Autorski alati (*Authoring Tools*) su namenjeni dizajniranju i kreiranju nastavnih sadržaja i obezbeđivanju interaktivnih svojstava (npr., prenos i procesiranje multimedijalnih podataka, proveravanje veština, primenjena vežbanja, studije slučajeva, simulacije, i

povezivanje naslova radi lakog pristupa bazi znanja, diskusionim grupama, i sinhronoj komunikaciji kao što je *chat*).

4.1 Arhitektura sistema

Arhitektura bazirana na nezavisnim alatima omogućava vrlo laku zamenjivost postojećih alata novima. Obezbeđivanje nove funkcionalnosti rešava se dizajnom novog alata. Kod samostalnih alata ne postoji direktna kooperacija. Oni jednostavno koriste zajedničke resurse (BP za podatke i fajl repozitorijumi za sadržaje). Slaba, zapravo nikakva sprega kod ovakvog arhitekturnog rešenja pored navedenih prednosti ima i značajne nedostatke. Nezavisni alati mogu da obezbede samo relativno jednostavne funkcionalnosti. Svako proširivanje, znatno usložnjava sam alat, i zahteva spregu više alata u obezbeđivanju željenih funkcija. S druge strane, delovi alata koji su namenjeni za komunikaciju prema BP i fajl sistemu (podacima i sadržajima), opterećuju resurse sistema domaćina (memoriju, OS, DBMS), jer se povećava broj funkcionalnih poziva između aplikacije alata i navedenih sistemskih resursa. Ovi pozivi povećavaju transparentnost sistema, što potencijalno može da naruši njegovu bezbednost.

Raznolikost korisničkih interfejsa, takođe predstavlja jedno od nedostataka ovog pristupa. Svaki alat ima sopstveno kreiran interfejs, tako da korisnici za svaku funkcionalnosti moraju da poznaju interfejs odgovarajućeg alata. Čak i slični alati, namenjeni za isti tip funkcionalnosti imaju vrlo različito dizajnirane korisničke interfejse. U praksi se pokazalo da je jednostavan interfejs, baziran na sloganu *sve na jednom mestu*, sa uniformnim pristupom i pravilima korišćenja, ključno za uspeh korisnički orijentisanih aplikacija.



Ilustracija 47: Arhitektura sistema

Da bi sistem mogao da ima mnogobrojne funkcionalnosti, implementirane standarde e-učenja, da bude prezentaciono i navigaciono adaptivan, a u isto vreme fleksibilan (da se lako modifikuje i proširuje) i skalabilan, da može da se integriše u IS domaćina, kao

njegov deo za edukativnu namenu, on mora da ima modularnu arhitekturu (Ilustracija 48). Za razliku od svih poznatih softverskih rešenja koja omogućavaju e-učenje korišćenjem javnog Interneta, koja rade na bazi klijent – server paradigme, pri čemu je gotovo sva aplikaciona logika na serverskoj strani (*fat server – thin client*), predloženo rešenje ima distribuiranu aplikativnu logiku i na klijentskoj i na serverskoj strani. Razvijene su posebne klijentske aplikacije za svaku od korisničkih uloga. Svaka klijentska aplikacija je dizajnirana da omogući pouzdanu komunikaciju sa serverom, da delimično procesira korisničke unose (validacija i konverzija podataka, kriptografska zaštita radi prilagođenja za slanje prema serveru) i da korisnicima prezentuje primljene serverske podatke na željeni način (vizuelizacija podataka, reprodukcija multimedijalnih sadržaja i sl.).

Serverska strana je kompleksna, jer predstavlja fuziju različitih komponenti, namenjenih različitim funkcionalnostima. Ove komponente se mogu posmatrati na dva načina:

- prema vrsti korisnika – studentska, nastavička i administratorska, i
- prema delu sistema koje opslužuju – komponente (servisi) koji su klijentski orjentisane, komponente (servisi) koje su orjentisane prema sistemu domaćina i komponente (servisi) koji su orjentisani prema modulima za e-učenje

Glavna namena serverskog dela je grupisanje svih servisa koje sistem može da ponudi korisnicima, kooperaciju sistema za e-učenje sa IS domaćina, i potpuno odvajanje interfejsa prema korisniku, aplikacione logike i skladišta podataka.

Nastavnička, studentska i administratorska aplikacija predstavljaju sistemske komponente – jezgro sistema za e-učenje. U ovim aplikacijama implementirane su sve funkcionalnosti e-učenja. Ove komponente koriste podatke iz IS domaćina posredstvom servis provajdera, imaju sopstvenu BP specijalizovanu samo za podatke e-učenja i repozitorijum za sadržaje e-učenja. U ovim aplikacijama realizovana je adaptivna funkcionalnost, napredno ocenjivanje, profilisanje i analizu rezultata studenata.

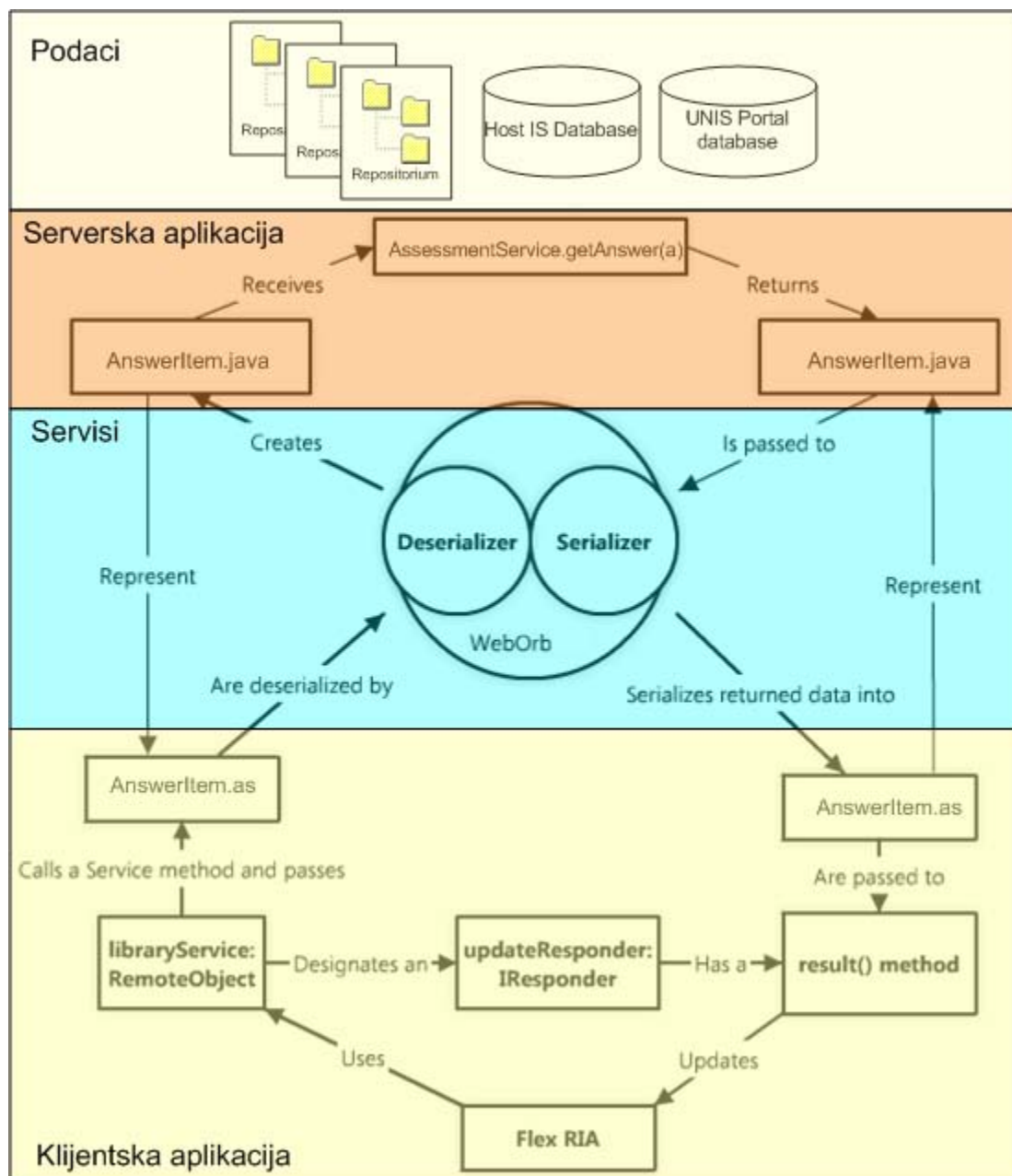
Podaci e-učenja su, kao što je već iskazano, u posebnoj BP u odnosu na IS domaćina. Pošto se u njoj čuvaju podaci e-učenja studenata, a u IS domaćina ostali podaci studenata, sistem treba da održava njihovu konzistentnost. Sinhronizacija podataka realizuje se na nivou servisa, implementiranih na servis provajderu.

4.2 Izbor tehnologija

Predloženo rešenje sistema ima četvoroslojnu arhitekturu (Ilustracija 49):

- Prezentacioni sloj – implementiran u klijentskim aplikacijama
- Sloj servisa – implementiran kao servis provajder
- Sloj poslovne logike – nastavnička, studentska i administratorska aplikacija
- Sloj podataka – sistem za upravljanje BP e-učenja i repozitorijum nastavnih sadržaja

Raslojavanjem se postiglo potpuno odvajanje prezentacije, logike, podataka i komunikacionog modela. Potpuna nezavisnost slojeva je omogućila primenu najadekvatnijih tehnologija za svaki sloj posebno.

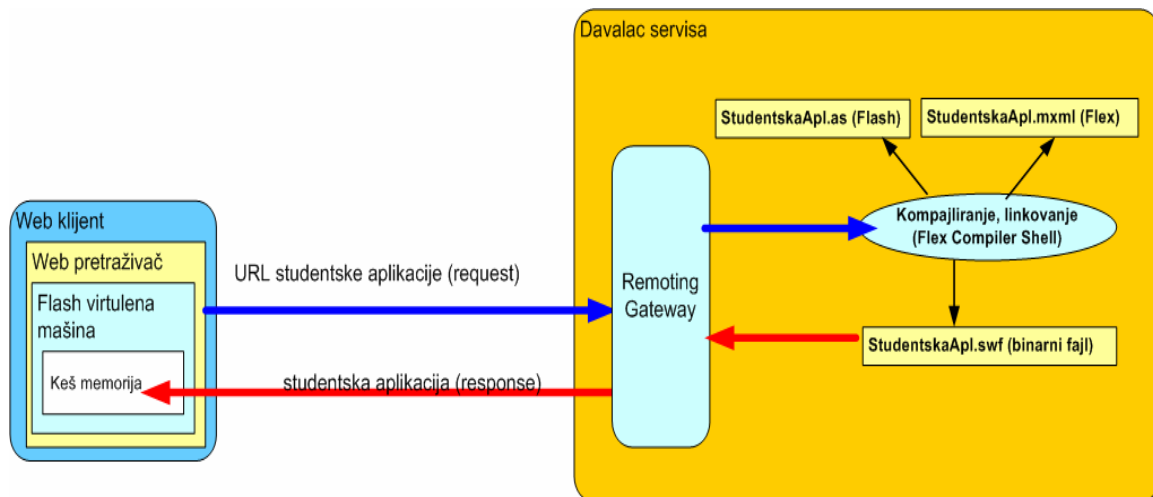


Ilustracija 48: Konceptualno rešenje komunikacije klijentskih aplikacija sa serverskim delom

Na prethodnoj slici (Ilustracija 49) je prikazano predloženo tehnološko rešenje naprednog sistema za e-učenje. Aplikaciona logika je distribuirana u sva četiri sloja. Svaki od njih ima specifične odgovornosti. Kao što je već objašnjeno, sloj podataka sadrži repozitorijume materijala i podatke e-učenja smeštene u BP. Sloju podataka e-učenja isključivo pristupaju aplikacije serverske strane, implementirane u Java programskom jeziku [89]. Na klijentskoj strani to su aplikacije implementirane u Flex 3 tehnologiji (poglavlje 4.2.1). Komunikacija između klijentskih i serverskih aplikacija je indirektna – preko servisa koje pruža WebORB servis provajder [90]. WebORB ima mogućnost fuzije aplikacija u različitim tehnologijama i programskim rešenjima, sa RIA aplikacijama na klijentskoj strani, koristeći Internet kao komunikacioni medijum (poglavlje 4.2.2).

4.2.1 Prezentacioni sloj

Prezentacioni sloj je u potpunosti implementiran na klijentskoj strani. Kod većine sistema za e-učenje, kod kojih se koriste Web tehnologije kao komunikacioni model, korisnički interfejs se dinamički generiše u vidu Web stranica, a zatim se isti distribuira klijentima. Poslata Web stranica se učitava u prozor Web pretraživača na strani klijenta. Svi LMS (poglavlje 3.5), AHES i ITS (poglavlje 3.4) opisani sistemi, distribuiraju na ovaj način sadržaje učenja. Nedostatak ovakvog pristupa je u ograničenim mogućnostima zaštite i prezentacije podataka. Za transport se koriste mehanizmi HTTP zahteva (*HttpRequest*) i odgovora (*HttpResponse*). Podaci koji se prenose su isključivo karakterski nizovi, tako da je sistem opterećen konverzijama objektnih tipova u osnovni. HTTP protokol je transparentan, a korišćenje HTTPS znatno uvećava opterećenje kanala. Ova karakteristika može da bude kritična u uslovima intenzivnog distribuiranja multimedijalnih materijala, što je osnovni slučaj korišćenja sistema za e-učenje. Najčešće pominjane tehnologije za realizaciju ovakvog koncepta su PHP [91], JSP [92], Servlet [93], ASP [94] i Python [95]. Skromne mogućnosti dizajniranja korisničkog interfejsa su delimično rešene korišćenjem AJAX [96] poziva u klijent – server komunikaciji, novih framework-a kao što su Spring [97], Struts [98], JSF [99] i jezika JavaScript [100] i ActionScript [101]. U predloženom rešenju, za izgradnju klijentskih aplikacija je iskorišćena je Adobe Flex 3 tehnologija [102]. Ona omogućava razvoj RIA (*Rich Internet Application*) – aplikacija sa bogatim interfejsom (Ilustracija 50).

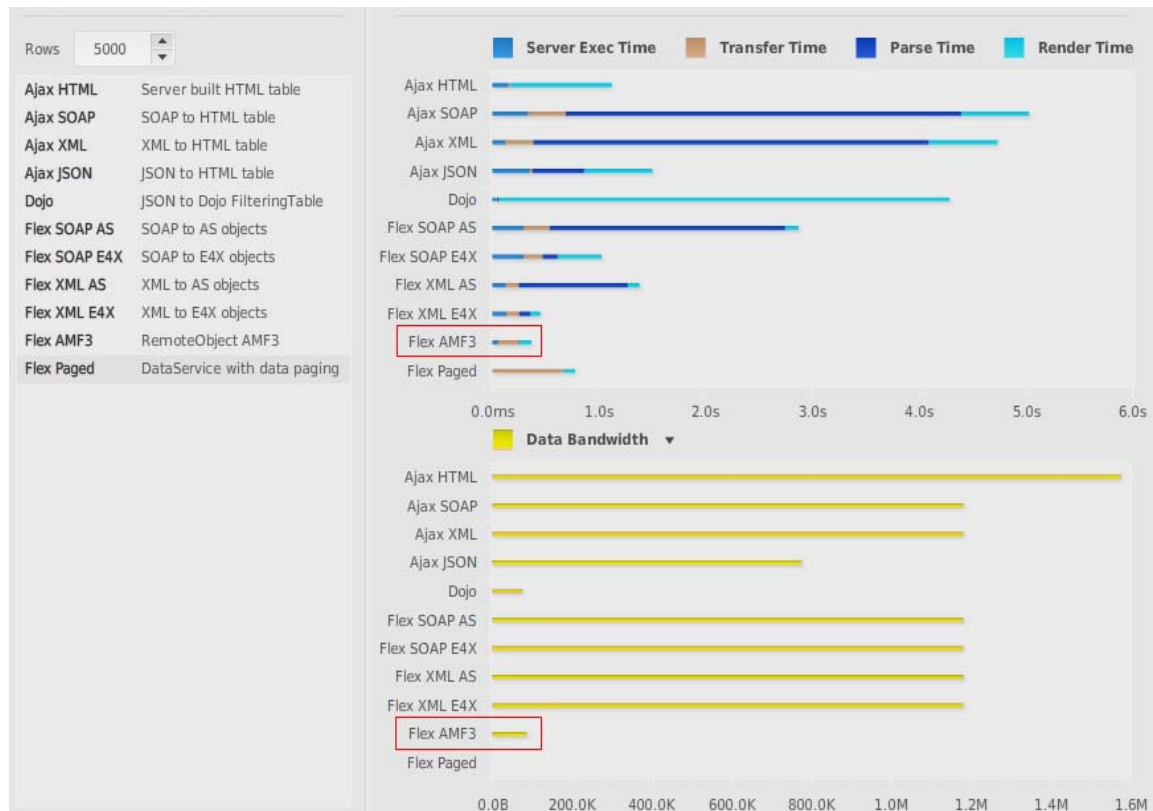


Ilustracija 49: Način pripreme i distribuiranja klijentske aplikacije

Kada korisnik izabere URL sistema za e-učenje, davalac servisa priprema klijentsku aplikaciju. Na osnovu ActionScript fajlova (implementacija funkcionalnosti) i MXML fajlova (implementacija interfejsa), pripremaju se veze ka serverskoj strani i obezbeđuju se inicijalni podaci i objekti. Davalac servisa zatim vrši kompajliranje i linkovanje. Kao rezultat ovog procesa, dobija se binarni SWF fajl, koji se šalje Web pretraživaču klijenta. Ovaj fajl se učitava u Web pretraživač. Nakon učitavanja, aplikacija preko Flash virtuelne mašine preuzima potpunu kontrolu nad pretraživačem i uspostavlja komunikaciju sa servis provajderom na bitskom nivou. Tada je klijentska aplikacija spremna za obradu zahteva korisnika. Pošto se komunikacija između klijentske aplikacije

i serverskog dela odvija razmenom objekata, poseban modul – *Remoting Gateway* namenjen je serijalizaciji i deserijalizaciji tih objekata.

Ovako pripremljena veza može da se ostvaruje na različite načine – transportovanjem preko HTTP protokola, ili sinhronim strimovanjem (obzirom da je prenos svakako na bitskom nivou).



Ilustracija 50: Upoređivanje performansi različitih tehnologija za RIA klijentske aplikacije

Pri izboru tehnologije za izgradnju klijentske aplikacije obavljeno je testiranje tehnologija kandidata. Na prethodnom dijagramu (Ilustracija 51), predstavljene su performanse odabrane tehnologije (Flex AMF3) i ostalih RIA tehnologija.

Obzirom da se radi o komunikaciji na nivou bita, ne postoje neinformativni sadržaji (zaglavlja, tagovi i sl.) tako da je zahtevan propusni opseg po kanalu redukovao. Pošto na klijentskoj strani nema potrebe za parsiranjem, ukupno vreme je zbir vremena pripreme, transfera i renderovanja sadržaja. Priprema fajlova je brza zahvaljujući korišćenju optimizovanog Flex 3 kompajlera. Na osnovu prikazanog, a u odnosu na ostale tehnologije, Flex AMF3 predstavlja najoptimalnije rešenje.

4.2.2 Sloj servisa

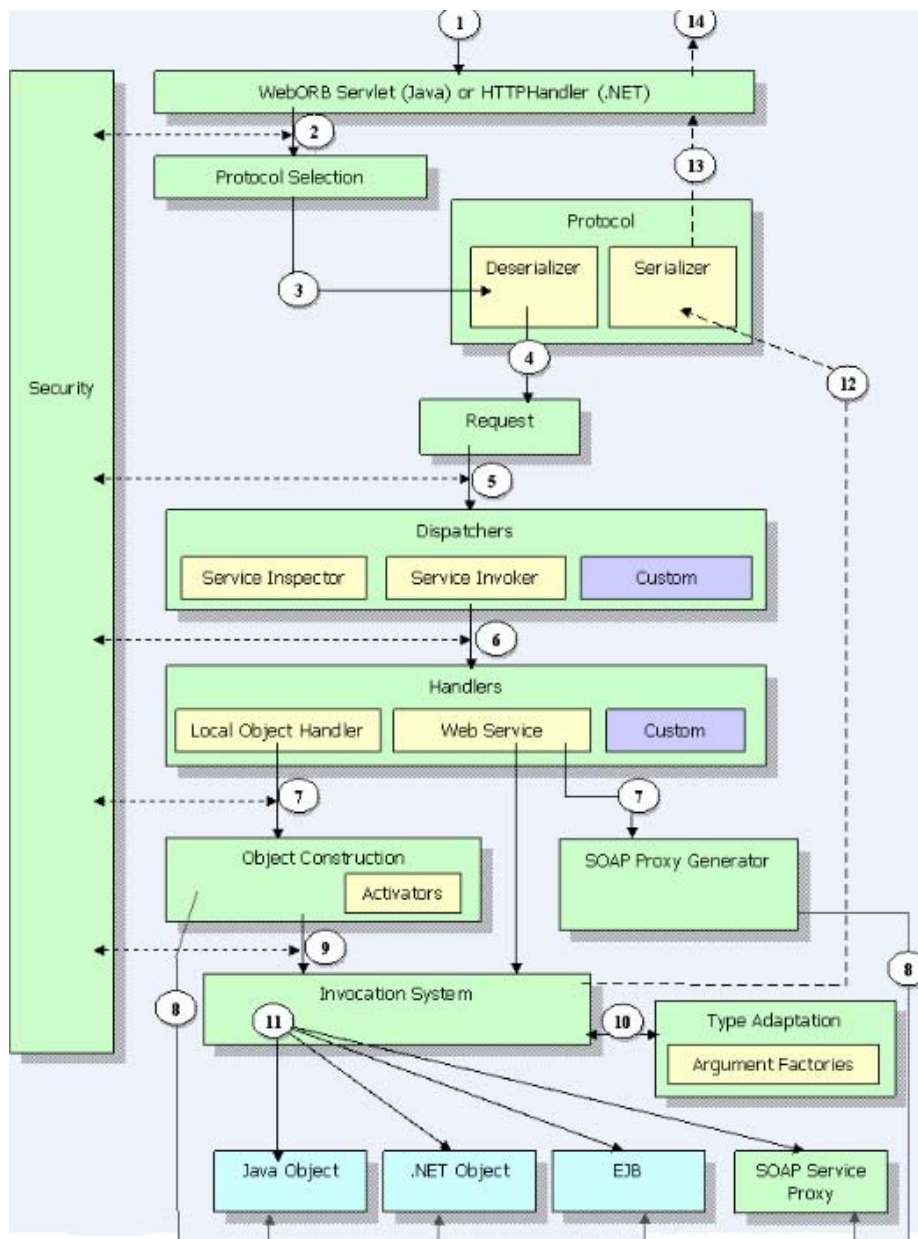
WebORB servis provajder ima raslojenu arhitekturu (Ilustracija 52). Osnovne komponente arhitekture su:

- Klase za rukovanje protokolima (*Protocol Handlers*) – podržani su različiti formati razmene poruka:

- o AMF binarni protokol, namenjen za Adobe Flash i Flex klijentske aplikacije
 - o WOLF (Web Object Literal Format) otvoreni protokol zasnovan na XML sintaksi za podršku AJAX klijenata.
- Klase dispečeri – vrše kategorizaciju zahteva primljenih od klijentskih aplikacija, po potrebi vrše filtriranje i prosleđuju zahteve odgovarajućim klasama rukovaocima.
- Klase rukovaoci – vrše ispitivanje servisa i uvlače metode koje treba da se obave. Za svaku tehnologiju postoje posebne grupe rukovaoca: za Java objekte, .NET objekte, SOAP web servise i EJB (*Enterprise Java Beans*).
- Bezbednosni modul – implementira bezbednosnu politiku: ograničava, ili dozvoljava pristup sistemskim resursima, servisima i metodama. Pre nego što se zahtev prosledi sledećoj komponenti u lancu procesiranja, ovaj modul slojevito proverava prava pristupa.
- Aktivatori omogućavaju kreiranje (instanciranje i inicijalizaciju) objekata.
- Radionica za objekte – namenjena je za instanciranje Java i .NET objekata. WebORB koristi konstruktore bez argumenata.
- Adapter tipova – pre funkcionalnog poziva (preciznije, uvlačenja metode objekta) ova komponenta analizira podatke koje je poslao klijent i konvertuje ih u očekivani tip.

Princip funkcionisanja predložene servisne arhitekture je sledeći. Kada klijentska aplikacija pošalje zahtev serverskoj, isti se prosleđuje odgovarajućem rukovaocu protokola, koji ga deserijalizuje. Prvo parsira zahtev, a zatim instancira objekat koji enkapsulira zahtev. Ovaj objekt ima naziv klase, naziv metode i listu argumenata. Ove objekte zatim preuzimaju dispečeri, koji ispituju dostupne servise i/ili uključuju dostupne metode.

Navedenim akcijama dispečera se realizuju klijentski zahtevi. Tokom njih se instanciraju odgovarajući objekti (u konkretnom slučaju Java objekti), pozivaju njihove metode, prilagođavaju i prosleđuju argumenti metodama, prihvataju rezultati metoda, a zatim adaptiraju u oblik pogodan za serijalizaciju, nakon čega se vrši slanje prema klijentima.



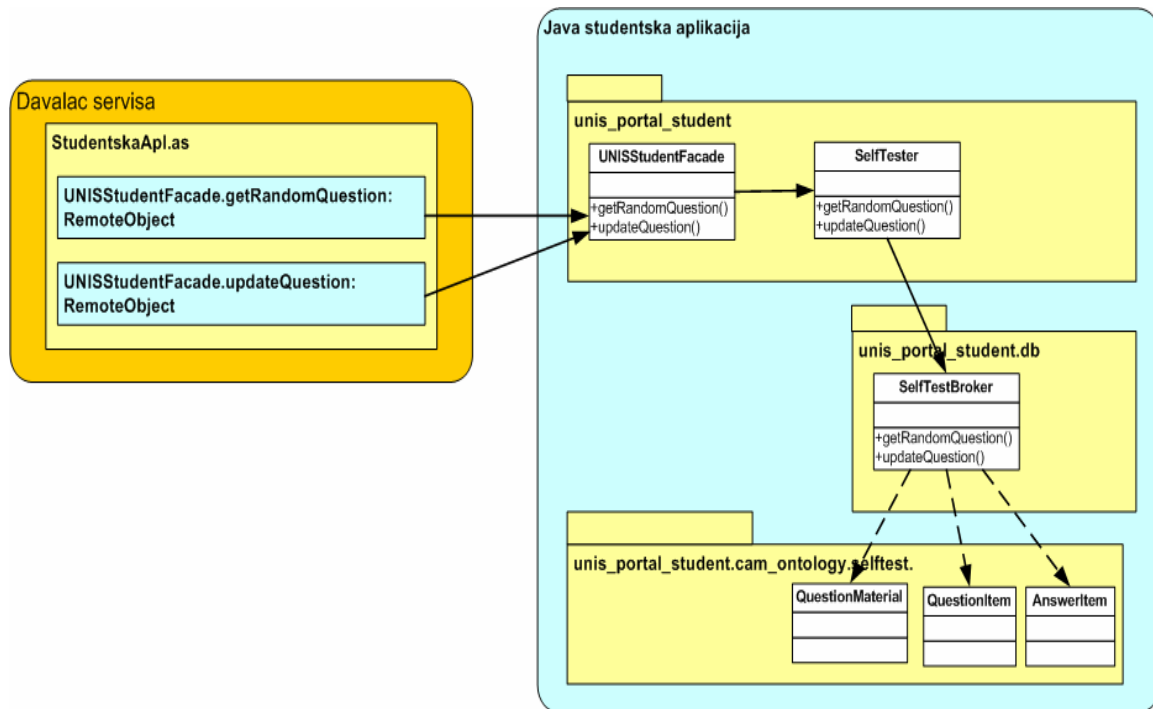
Ilustracija 51: Arhitektura WebORB davanja servisa

Navedena platforma – davalac servisa u potpunosti zadovoljava zahteve specificirane u opisu sistema:

- Integracija RIA i serverskih tehnologija posredstvom servisa.
- Podrška protokolima između klijenata i servera na nivou bita.
- Mogućnost proširivanja dodavanjem odgovarajućih rukovaoca objekata za npr RMI registry objekte, ili COM/OLE objektno povezivanje.
- Mogućnost primene i prilagođavanja bezbednosne politike na različitim aplikacionim i komunikacionim nivoima.
- Platformska nezavisnost – davalac servisa je oslonjen na Java virtualnu mašinu

4.2.3 Sloj poslovne logike

Sloj poslovne logike je implementiran kao tri posebne Java aplikacije. Isto kao i serverska platforma, aplikacije su platformski nezavisne i predstavljaju pakete klasa. Ove aplikacije su nevidljive za klijente. Klijenti pristupaju direktno servisima, a servisi se obraćaju klasama Java aplikacija. Koncept serverske platforme koji omogućava posredovanje je klasa *RemoteObject* (Ilustracija 53).



Ilustracija 52: Povezivanje sloja poslovne logike i servis provajdera

Postoje dva pristupa u povezivanju. Instanciranje jednog *RemoteObject* objekta po jednoj korišćenoj java klasi, ili instanciranje *RemoteObject* objekata za svaki funkcionalni poziv unutar svake korišćene java klase. Nedostatak U predstavljenom rešenju, opredeljenje je za drugu opciju. Pošto serverska platforma komunicira samo sa jednim Java objektom (klasa *UNISStudentFacade*), to znači da postoji jedna instanca ove klase za svaku aktivnu klijentsku aplikaciju (sesiju). Da bi se omogućilo pouzdanije rukovanje izuzecima, za svaku metodu ove klase se kreira jedan *RemoteObject*, za koji se vezuju posebni i po potrebi pojedinih metoda – specifični rukovaoci događajima. Na strani Java aplikacije *UNISStudentFacade* objekat distribuira odgovornost prosleđivanjem funkcionalnog poziva kompetentnim objektima (prema UML pattern-ima *Ekspert* i *Kreator* [113]) u istom ili drugim paketima aplikacije.

4.2.4 Sloj podataka

Na sloju podataka čuvaju se podaci potrebni za e-učenje:

- Metapodaci sadržaja za učenje (nazivi, ključne reči, kategorija, referenca)
- Metapodaci strukture predmeta (hijerarhijska struktura činilaca predmeta)
- Podaci studenata vezani za učenje (rezultati, log fajlovi, navigacione putanje i sl.)

- Metapodaci i podaci za proveravanje (kriterijumi za ocenjivanje, reference na materijale pitanja i odgovora, definicije tipova proveravanja, pitanja i odgovora, itd.)
- Podaci za adaptaciju (tipovi šablona za prezentaciju, kriterijumi premisa i konsekvenci pravila za zaključivanje, pedagoške strategije i taktike itd.)

Za smeštanje ovih podataka opredeljenje je sistem za upravljanje relacionom bazom podataka (SUBP, eng.RDBMS) – MySQL [103]. Razlog korišćenja MySQL-a je da podržava najveći broj platformi (OS), ima maksimizirane karakteristike u pogledu veličina tipova polja za tekstualne podatke, blob-ove, podršku za različite tekstualne formate [104]. Takođe ima dobre performanse u kontroli pristupa i primeni bezbednosnih politika. Pored toga, ovaj SUBP je besplatan.

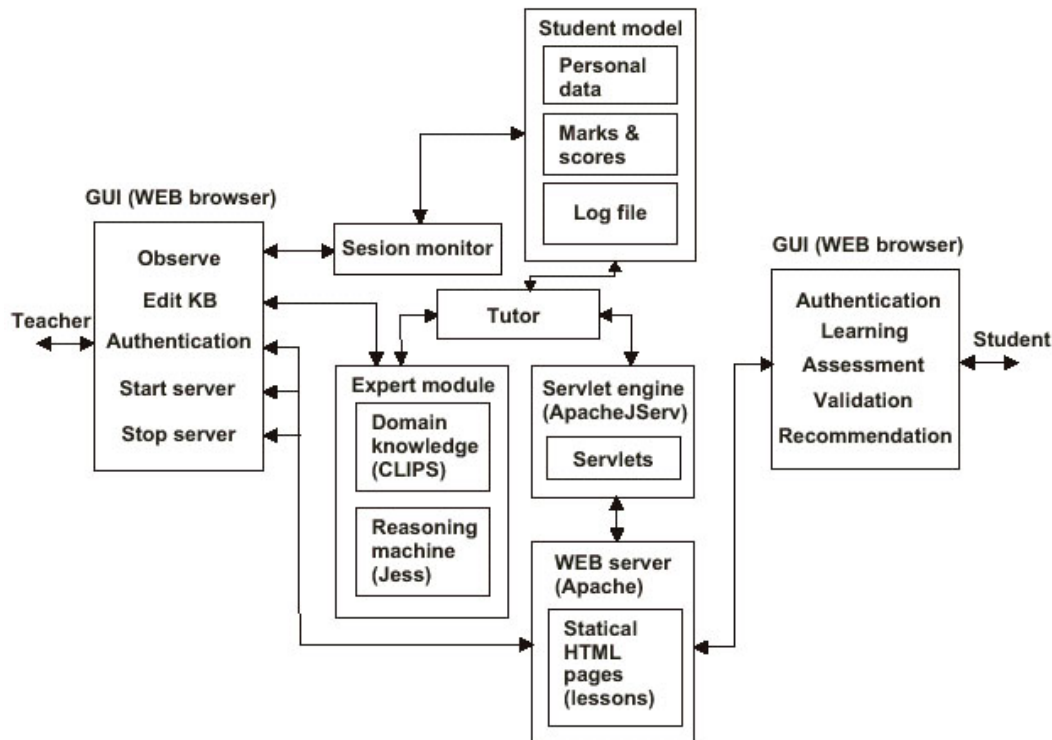
Sadržaji elektronskog učenja takođe pripadaju ovom sloju. U tu svrhu su obezbeđeni posebni direktorijumi za smeštanje resursa učenja i testiranja. Fajlovima se isključivo pristupa preko aplikacije, tako da klijentske aplikacije ne vide ni putanje ni prave nazive fajlova. Razmatrana je opcija da se fajlovi čuvaju kao zapisi u tabelama SUBP (korišćenjem blob polja), a aplikacija da koristi tzv.virtuleni fajl sistem. Međutim performanse sistema se jako narušavaju dodatnim procesom regenerisanja fajla na osnovu podataka iz BP. Jednostavno, OS taj proces (iščitavanje/upis fajlova na spoljnu memoriju) izvrši brže nego SUBP, koji pristupa ciljnoj BP (koja inače predstavlja fajl za sebe), ekstrahuje podatke, konvertuje ih i rekonstruiše originalni fajl. Sadržaji raspoređeni na fajl sistemu, zajedno sa BP u kojoj se čuvaju njihovi metapodaci, čine grade jedinstveni sistemski repozitorijum.

4.3 Prethodna istraživanja

Izradi aktuelnog sistema prethodila su istraživanja (projekti) vezani za e-učenje. Svaki projekat je istraživao nove funkcionalnosti i istovremeno bio proširenje prethodnog. Prvi od njih je inteligentni tutorski sistem *Code Tutor*.

4.3.1 Code Tutor

Code Tutor je mali tutorski sistem zasnovan na Web tehnologijama [105]. Namena sistema bila je brza priprema studenata u domenu radio-komunikacija. *Code Tutor* ima tipičnu ITS (vidi „Skratice“) arhitekturu (Ilustracija 54). Tutorsku komponentu predstavlja pedagoški modul. Baza znanja (domensko znanje) je kodirano produkcionim pravilima. *Jess* [106] shell za ekspertske sisteme je korišćen kao modul za zaključivanje. Podaci studenata, kao što su podaci prikupljeni kroz sesije, rezultati, preference, smešteni su u eksplicitnom modelu studenta.



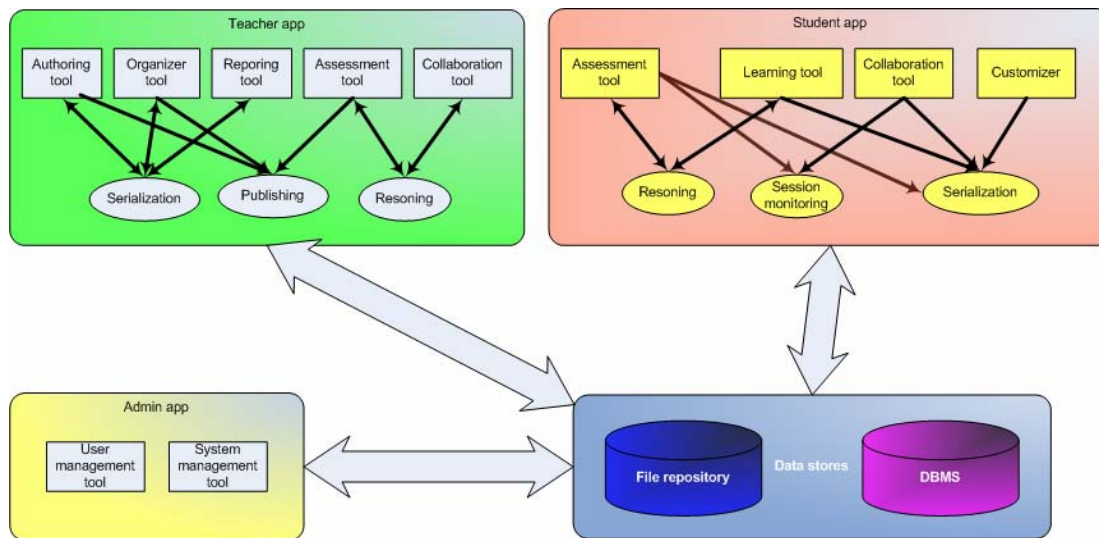
Ilustracija 53: Arhitektura Code Tutor-a

Komunikacioni model je predstavljen dinamički kreiranim Web stranicama. Jezgro *Code Tutor-a* je modul *Tutor*. Ovaj modul procesira aktuelne podatke studenta korišćenjem pravila. Dobijeni zaključci se koriste za određivanje vrste pomoći koju aplikacija treba da pruži konkretnom studentu. Na primer, to može da bude promena mogućnosti navigacije, (promena redosleda sadržaja, pristupačnost sadržaja bazirana na stanju modela studenta i sl.), adaptacija provera (višenivojski testovi) i preporuke za bolji uspeh (u formi internih linkova).

Bez obzira što se radi o izolovanom sistemu, domenski usko fokusiranoj aplikaciji, bez mogućnosti razmene podataka sa okruženjem, nastavničko opredeljenje bilo je da se sistem pokazao korisnim, i da su studenti bili zadovoljni ovim načinom učenja. Pored inteligentnog ponašanja, važno unapređenje u *Code Tutor-u* je nalaženje načina automatske promene pravila u bazi znanja, a na osnovu nastavničkih unosa. (menjanje strukture predmeta, kriterijuma proveravanja, i relacija sadržaja). Promenom u pravilima, proizvodile bi se promene u korisničkom interfejsu i ponašanju sistema.

4.3.2 Multitutor

Ovaj sistem predstavlja proširenje prethodnog sistema [109]. *Multitutor* je skup alata zasnovan na Web tehnologijama, dizajniran i za nastavnike i za studente. Veliki broj funkcionalnosti *LMS* (vidi „Skratčenice“), kao i naponi zajednice za e-učenje u vezi standardizacije bili su glavni motivi za dizajn novog sistema. Arhitektura novog sistema je zasnovana na modelu komponenti (Ilustracija 55).



Ilustracija 54: Arhitektura LMS sistema Multitutor

Sistem se sastoji od tri dela (aplikacija): nastavnička, studentska i administratorska aplikacija. *Multitutor* je namenjen da bude domenski nezavisan sistem sa mnogo više upravljačkih funkcionalnosti. Iz tog razloga su administratorske funkcije izdvojene u posebnoj aplikaciji. Uloga administratora je vrlo bitna u procesu učenja: odgovoran je za podešenost i funkcionisanje sistema (npr. nazivi aplikacionog, Web servera i servera BP, korišćeni logički portovi, upravljanje repozitorijumima sadržaja, itd.), za upravljanje korisničkim nalogima (nastavnici, saradnici, studenti, studentske grupe), podešavanje i održavanje lokalizacije sadržaja (ažuriranje rečnika korišćenih za namene lokalizacije).

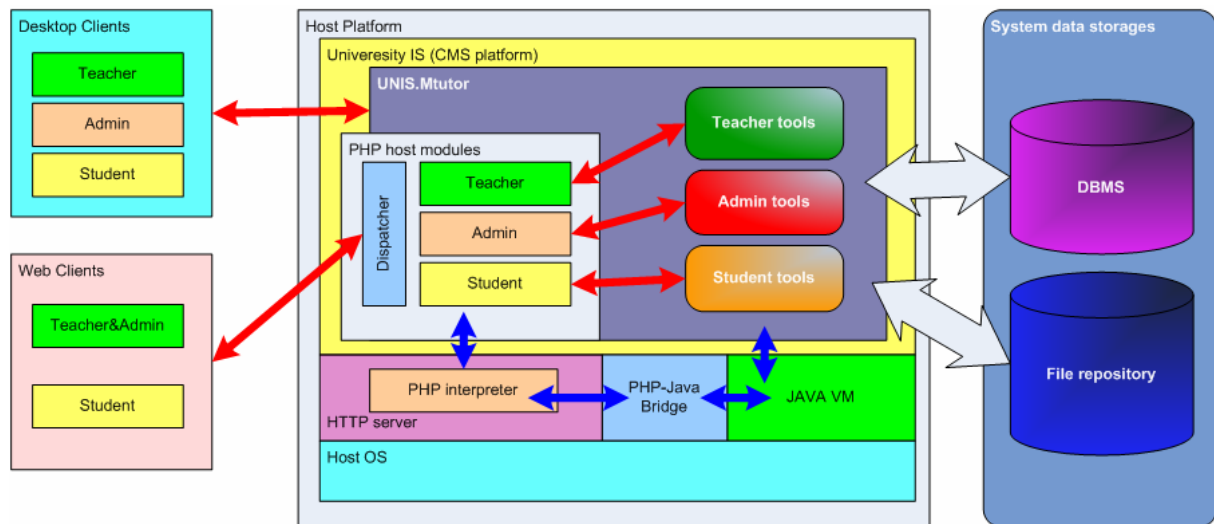
Slično *LMS*-ovima, arhitektura *Multitutor*-a se može razmatrati kao raslojena. Aplikacije su komponovane od različitih alata, koji su uzajamno nezavisni. Sistem se ponaša kao skup alata (*framework*), u kome su alati zapravo komponente koje obezbeđuju različite vrste servisa korisnicima. Alati interaguju sa sistemom kroz tzv. systemske procese. Za razliku od *CodeTutor*-a podaci su izdvojeni (baza znanja, korisnički podaci, metapodaci kurseva, podaci saradnje i sadržaji učenja). Na taj način se obezbedilo da je moguće njihovo višestruko korišćenje u različitim funkcionalnim kontekstima *Multitutor*-a. Ovakav model omogućava laku modifikaciju i dodavanje novih komponenti u sistem.

Multitutor je dizajniran da sadrži neograničen broj kurseva. Nastavnicima je prepuštena odgovornost u vezi organizacije kurseva i korišćenja sadržaja učenja. *Multitutor* omogućava studentima da uče, testirajući ih povremeno i dajući im preporuke kako i šta da uče za bolji uspeh. Sistem vodi studente kroz resurse na različite načine, u zavisnosti od istorijskih podataka i aktuelnih rezultata, zapamćenih u studentskom modelu. Ove odluke su rezultat procesiranja korišćenjem baze znanja implementirane kroz produkciona pravila.

4.3.3 UNIS.Multitutor

Poslednje istraživanje je imalo fokus na proveravanju znanja studenata. Kao njegov rezultat, kreiran je *UNIS.Multitutor* [110] – sistem za proveru znanja. Takođe je dizajniran kao modularan serverski sistem kome se pristupa na dva načina: preko Interneta i Web pretraživača, i kroz lokalnu mrežu posredstvom desktop aplikacije.

Slično *Multitutor*-u, sistem je sastavljen od različitih alata grupisanih u tri aplikacije (nastavnička, administratorska i studentska).



Ilustracija 55: Arhitektura *UNIS.Multitutor* sistema

Sistem egzistira kao ugnježđen u aplikaciju domaćina, platformski je nezavisan (Ilustracija 56): PHP tehnologija je korišćena za implementaciju komunikacionog modela kod Web aplikacije, dok su preostali delovi realizovani kao paketi Java klasa (desktop interfejs, aplikaciona logika, brokeri prema BP itd.). Interakcije između PHP modula i Java klasa su realizovane posredstvom PHP-Java bridge komponente [111] (plavi tokovi).

Pored unapređenja u dizajnu aplikacije, smanjivanjem sprege između komponenti, značajan napredak kod *UNIS.Multitutor*-a je učinjen i na istraživanju mogućnosti povezivanja različitih tehnologija u jedinstven sistem, i utvrđivanju novih metoda u evaluaciji sistema. Naročit doprinos predstavljaju iskustva stečena u kreiranju testova, definisanju pedagoških strategija i tehnika proveravanja.

5. Nastavnička aplikacija

U narednim poglavljima opisani su najvažniji detalji iz dizajna i implementacije nastavničke aplikacije, sa akcentom na rešavanje naprednog ocenjivanja studenata i povezivanja sadržaja radi adaptivne navigacije i personalizacije sadržaja.

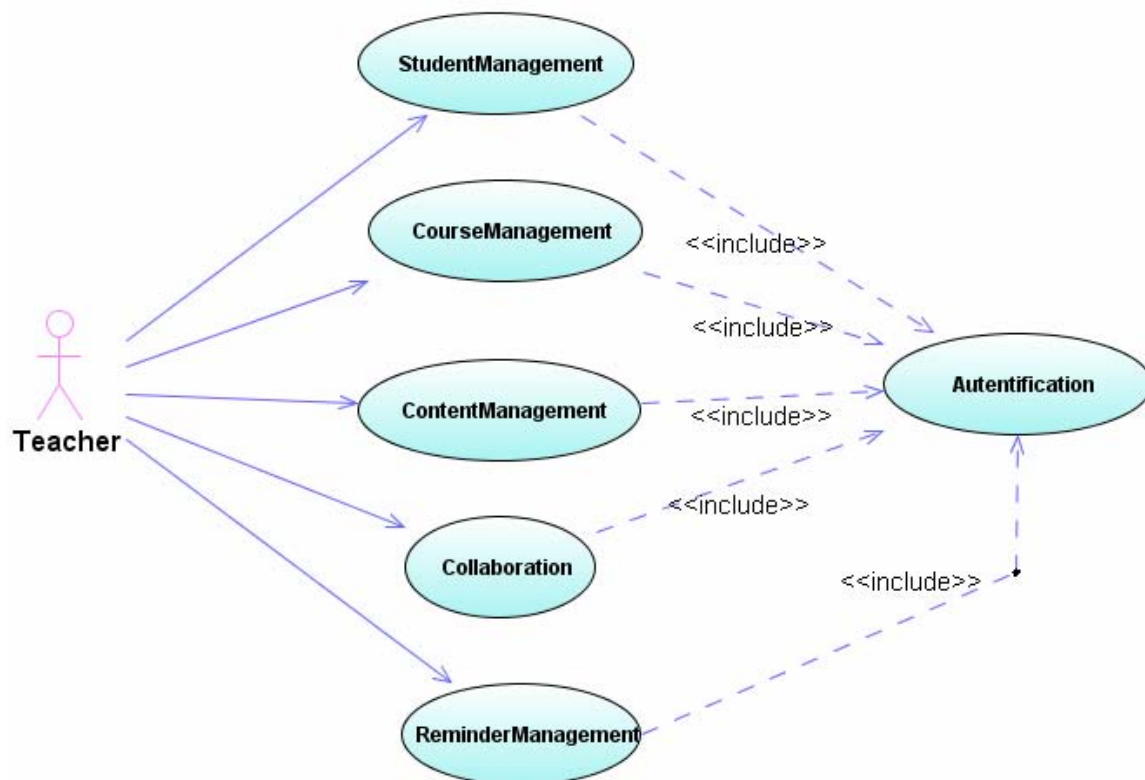
5.3 Dizajn i implementacija

U sledećim paragrafima opisani su u potpunosti slučajevi korišćenja nastavničke aplikacije, paketi klasa koji ju sačinjavaju i konceptualni dijagrami svakog paketa ponaosob sa težištem na relacijama između koncepata. Dijagramima aktivnosti predstavljen je korisnički aspekt, a dijagramima interakcije predstavljeni su detalji implementacije fuzzy logike u naprednom ocenjivanju.

5.3.1 Slučajevi korišćenja

U nastavničkoj aplikaciji postoji pet slučajeva korišćenja 0-tog nivoa (Ilustracija 57)., od kojih su četiri opciona i jedan obavezan:

- Upravljanje predmetima (*Course Management*) – ovaj SK obuhvata funkcionalnosti organizovanja nastave (definisanje strukture predmeta).
- Upravljanje sadržajem (*Content Management*) – iz aspekta *LMS* ovaj SK obuhvata glavne nastavničke funkcionalnost. Bez mogućnosti kreiranja i publikovanja sadržaja (na sistemu), sistem ne bi mogao da ostvari niti jednu drugu funkcionalnost.
- Upravljanje studentskim podacima (*Student Management*) – ovaj SK omogućava nastavnicima uvid u profil i napredovanje studenata kroz nastavni proces.
- Saradnja (*Collaboration*) – ovaj SK obuhvata komunikacione funkcionalnosti sistema.
- Autentifikacija (*Autentification*) – pošto korisnici rade na klijentskim platformama, a manipulišu podacima koji se nalaze na serverskoj strani aplikacije, sistem je obavezan da proveri identitet korisnika u svakoj njegovoj akciji. Na taj način sistemski resursi se štite od upada neautorizovanih korisnika.

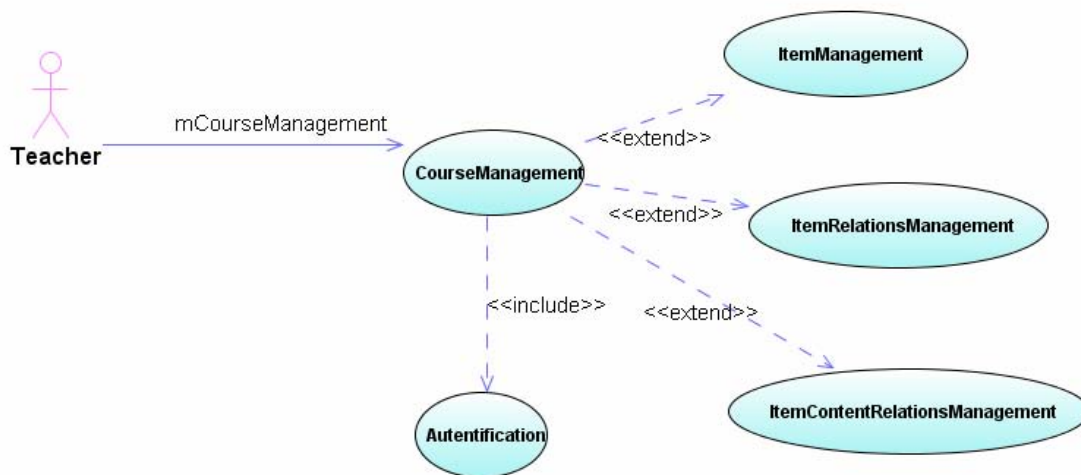


Ilustracija 56: Slučajevi korišćenja 0-tog nivoa nastavničke aplikacije

Svi slučajevi korišćenja (0-tog nivoa) nastavničke aplikacije su selektivni – uzajamno nezavisni. Svaki od njih predstavlja grupu funkcija, koje se ostvaruju nezavisno, tako da dalji razvoj dizajna omogućava kreiranje samostalnih alata za različite funkcionalnosti.

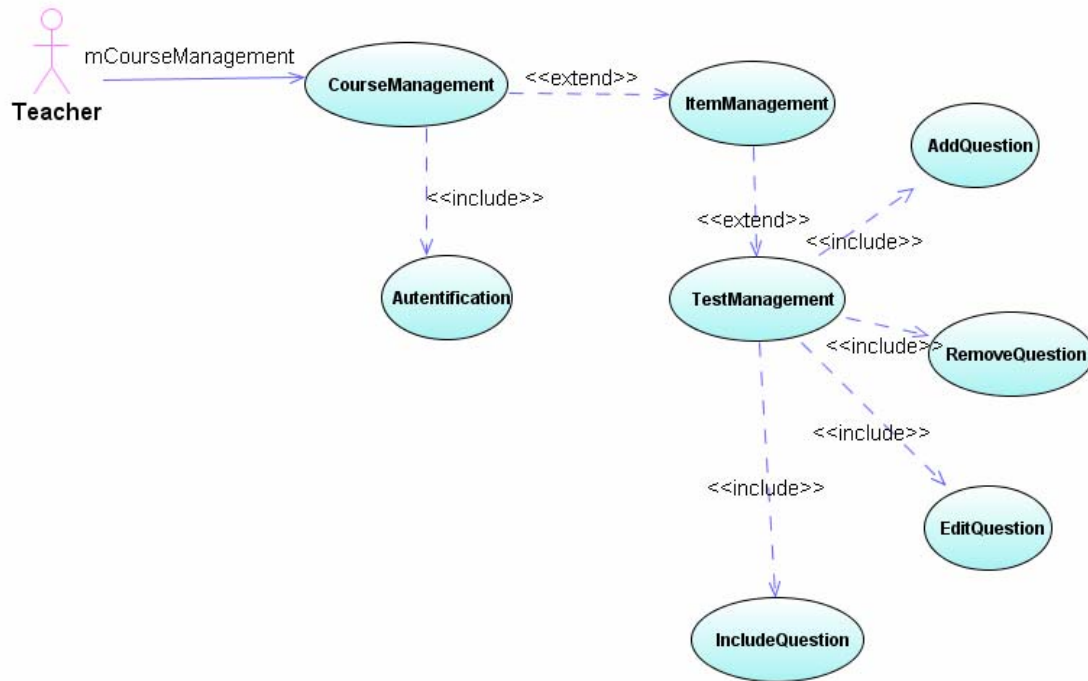
5.3.1.1 Upravljanje predmetima

Funkcionalnost upravljanja predmetima (SK 0-tog nivoa *CourseManagement*) se dalje dekomponuje (proširuje) na tri grupe funkcija, koje su predstavljene slučajevima korišćenja 1. nivoa (Ilustracija 58). Činioci (*Items*) predstavljaju osnovni element strukture – organizacije predmeta (koncept je u potpunosti preuzet iz IMS standarda za organizaciju sadržaja - poglavlje 3.2.3). Činilac je apstraktni koncept koji nije upotrebljiv u edukaciji sve dok se ne poveže sa drugim činiocima i sadržajima učenja.



Ilustracija 57: Nastavnička aplikacija – SK 1. nivoa za upravljanje predmetima

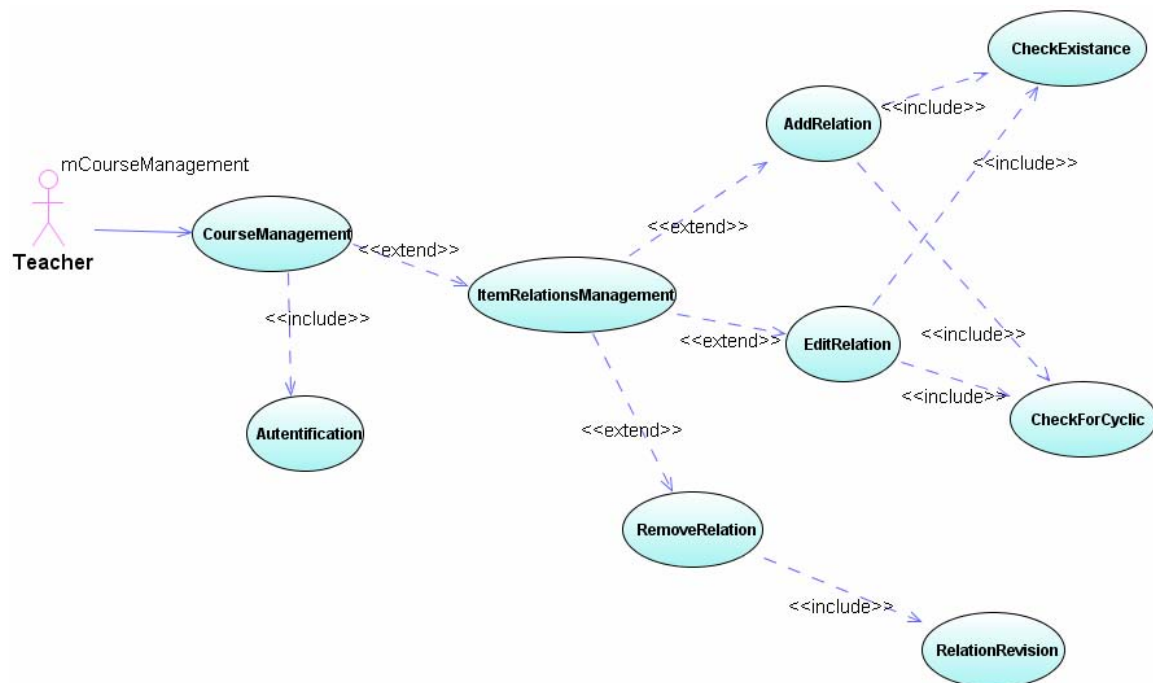
Slučajevi korišćenja 2. nivoa za upravljanje činiocima predmeta nisu predstavljeni jer obuhvataju primitivne funkcije dodavanja, uklanjanja i editovanja činilaca (slučajevi korišćenja *Add/Remove/Edit Item*), izuzev proširenja za upravljanje proverama (Ilustracija 59).



Ilustracija 58: Nastavnička aplikacija – SK 2. nivoa za upravljanje proverama

Za definisanje provera su usvojeni koncepti *IMS QTI* specifikacije (poglavlje 3.2.3). Na osnovu ovog standarda provere i pitanja obuhvaćena proverama, predstavljaju specifične činioce (*Item-e*) predmeta. Provere se sastoje od pitanja, koja mogu da budu različitog tipa (npr. sa ponuđenim odgovorima, sa slobodnim unosom odgovora, sa uparivanjem, itd.). Pitanja kao činioci u svojoj strukturi sadrže odgovore, ocene, opise načina procesiranja i povratne informacije za korisnika. SK *TestManagement* predstavlja specifični slučaj upravljanja činionicima. On obuhvata manipulaciju pitanjima (dodavanje, uklanjanje, editovanje), i uključivanje pitanja u proveru.

Upravljanje vezama između činilaca predmeta (*ItemRelationsManagement*) je kompleksnije, jer vezama se praktično ostvaruje usmereni graf u kome su čvorovi činioci a relacije lukovi (poglavlje 3.2). Upravljanje vezama se dekomponuje na tri slučaja korišćenja: za dodavanje (*AddRelation*), uklanjanje (*RemoveRelation*) i editovanje (*EditRelation*) veza. Dodavanje i editovanje obavezno uključuje dva dodatna slučaja korišćenja: proveru postojanja relacije (*CheckExistance*), i proveru grafa na cikličnost (*CheckForCyclic*).



Ilustracija 59: Nastavnička aplikacija – SK 2. nivoa za upravljanje relacijama predmetnih činilaca

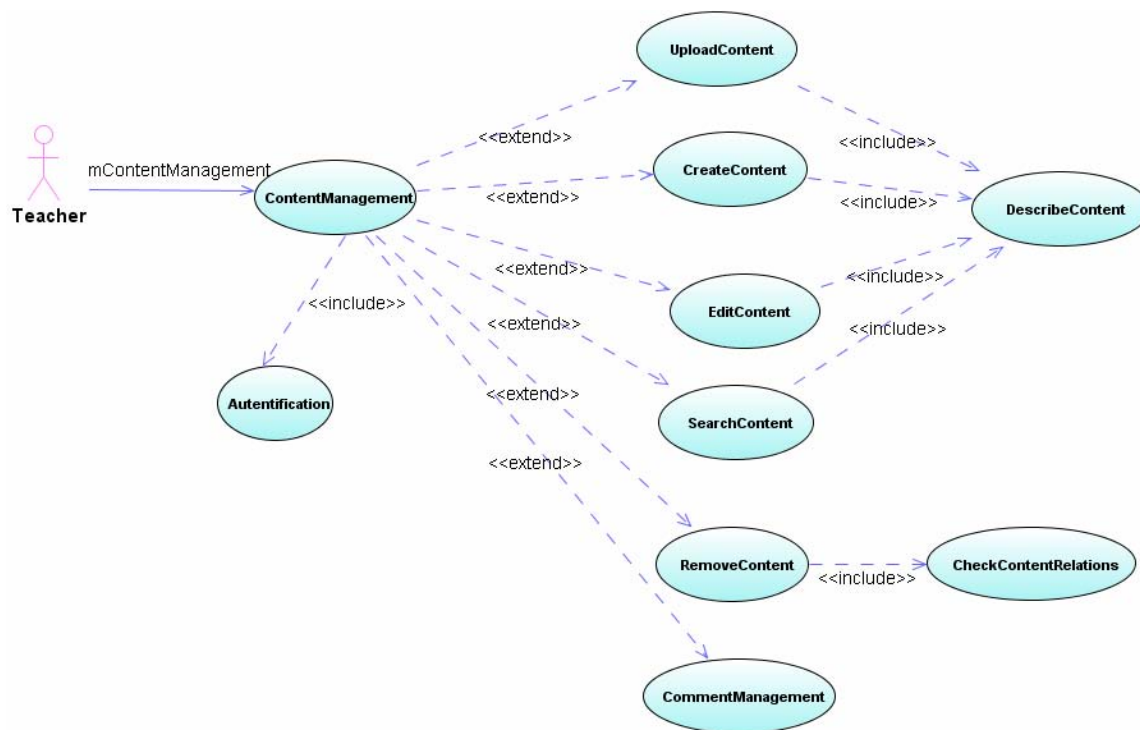
Navedene funkcije se dalje ne komponuju kroz ovaj tip dijagrama, a predstavljanje na dijagramima 1. nivoa bil bilo nepregledno (poglavlje 5.5).

5.3.1.2 Upravljanje sadržajima učenja

U savremenim *LMS*-ovima, upravljanje sadržajima učenja predstavlja jednu od najvažnijih funkcionalnosti. Čitav sistem korišćenja *LMS* gradi se oko sadržaja učenja i najviše zavisi od načina na koji su sadržaji struktuirani i opisani. Postoji sedam slučajeva korišćenja .1 nivoa za upravljanje sadržajima:

- Postavljanje sadržaja na sistem (*UploadContent*) – ovaj SK omogućava nastavniku da materijal sa klijentskog računara, postavi posredstvom mreže na server, predviđen za smeštaj sadržaja učenja. Pri tome sistem proverava na osnovu naziva, vremena kreiranja i veličine fajla da li sadržaj već postoji. Inače originalni nazivi fajlova nalaze pohranjuju se u BP, dok se svakom sadržaju dodeljuje novi (jedinostveni identifikator).
- Direktno kreiranje sadržaja u sistemu (*CreateContent*) – pored postavljanja (*upload-a*) sistem treba da omogući direktno kreiranje sadržaja, u formi tag-ovanog teksta (npr.HTML), koji u sebi može da ima ugnježdene i druge resurse.
- Editovanje sadržaja (*EditContent*) – ovaj SK omogućava editovanje direktno kreiranog sadržaja i opisa resursa (naziv, domen i ključne reči, kreator, tip).
- Pretraživanje sadržaja (*SearchContent*) – pretraživanje sadržaja se vrši na osnovu njihovih metapodataka – naslov, ključne reči, tip, domen, kreator.
- Uklanjanje sadržaja (*RemoveContent*) – ovaj SK je izvodljiv samo ako se specificiran resurs nije referenciran iz činilaca predmeta.

- Upravljanje komentarima (*CommentManagement*) – ovaj SK odnosi se na dodavanje, pregled, editovanje i uklanjanje komentara vezanih za konkretan sadržaj učenja. Komentarima je predstavljen mehanizam anotacije u sistemu. Kroz komentare, nastavnik može da da studentima sugestije, pomoć , kao i neke dodatne informacije vezane za sadržaj.
- Opis sadržaja (*DescribeContent*) – ovaj SK namenjen je da se resursu dodaju metapodaci (naslov, ključne reči, tip, domen, kreator), korisni za pretraživanje repozitorijuma.
- Provera veza sadržaja sa činionicima predmeta (*CheckContentRelations*) –treba da zaštiti integritet pripremljenih kurseva (predmeta). U slučaju da nastavnik želi da ukloni sadržaj, sistem mora da proveri da li se sadržaj već koristi (da li je povezan za činioce predmeta). Sistem dozvoljava da se sadržaj ukloni samo ako nije uključen u proces učenja (nema vezu ni sa jednim činiocem predmeta).



Ilustracija 60: Nastavnička aplikacija – SK 1. nivoa za upravljanje sadržajima učenja

Svi slučajevi korišćenja izuzev za uklanjanje sadržaja uključuju opisivanje sadržaja. Opisivanjem sadržaja, nastavnik unosi metapodatke o sadržaju: naziv, ključne reči, oblast na koju se sadržaj odnosi, tip sadržaja i druge standardne podatke (poglavlje 3.2.2).

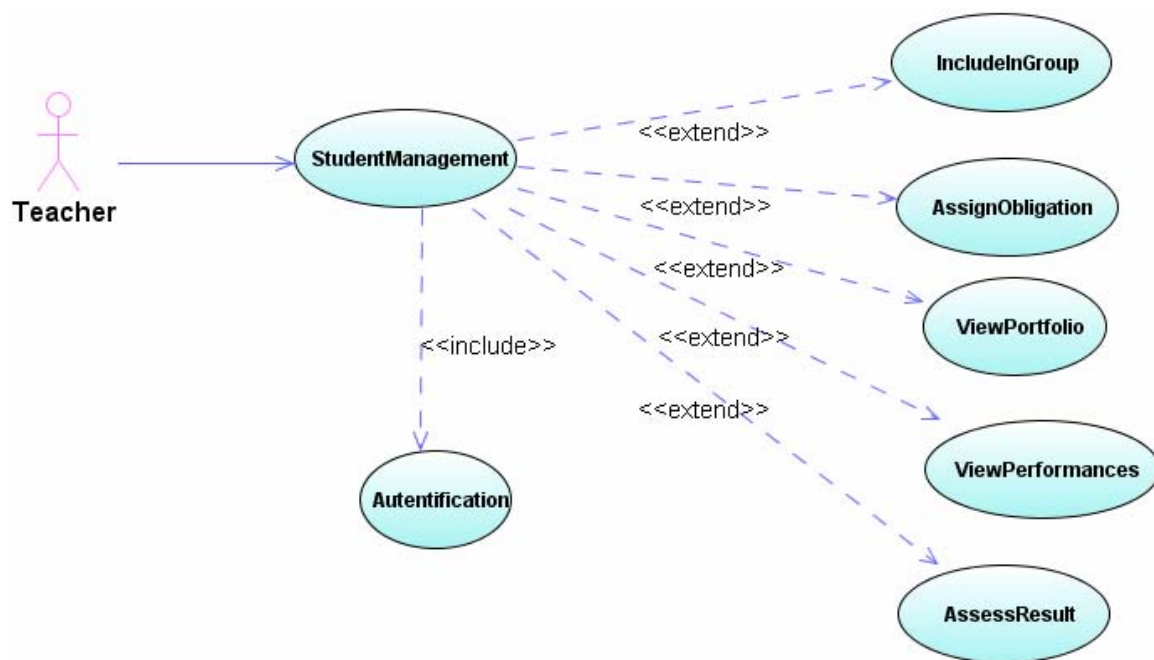
5.3.1.3 Upravljanje studentskim podacima

SK *Upravljanje studentskim podacima* specijalizuje u pet slučajeva korišćenja (Ilustracija 62):

- Uključivanje studenta u grupu (*IncludeInGroup*) – vrlo važan SK je uključivanje studenta u grupu, jer tek tada, student koji inače egzistira u sistemu, ima mogućnost pristupanja resursima učenja. Ovaj slučaj može biti automatski (npr.

implicitno se podrazumeva da se student priključuje određenoj grupi na osnovu podataka o fakultetu i godini studija), ili eksplicitno nastavnik priključuje studenta virtuelnoj grupi, radi ispunjenja grupnih obaveza, u kontekstu neke aktivnosti učenja.

- Dodela obaveza (*AssignObligation*) – Dodela obaveza je SK kojim nastavnik može da definiše obavezu (zadatak) na pojedinačnom ili grupnom nivou, pri čemu isti dodeljuje studentu, ili grupi.
- Pregled portfolija studenta (*ViewPortfolio*) – ovaj SK obuhvata pregled svih podataka studenta, merodavnih za konkretan predmet. To su podaci koji su obuhvaćeni studentskim modelom. Pored toga, obuhvaćeni su i izvedeni podaci, dobijeni statističkom obradom i zaključivanjem iz log zapisa studentovih sesija sa sistemom. U portfoliju su obuhvaćeni aktuelni sertifikati i licence koje student poseduje, radovi koje je student objavio, interesovanja i van-nastavne aktivnosti studenta.
- Pregled performansi studenta (*ViewPerformances*) – performanse obuhvataju sve izmerene vrednosti parametara po kojima se vrednuje studentski rad i rezultati. To mogu biti izvedeni podaci kao što su konkretne ocene, ali i opisi o efikasnosti u smislu tačnosti, brzini, vremenu, aktivnosti, ili prisutnosti na nastavi, ispunjavanju obaveza.
- Ocenjivanje rezultata (*AssessResult*) – pored mogućnosti automatskog ocenjivanja studenata (npr. u slučaju testiranja), nastavnik može da eksplicitno unosi ocene studenata u kontekstu konkretnog predmeta (npr. ocene seminarskog rada, ocene praćenja sa stažiranja, praksisa i sl.).

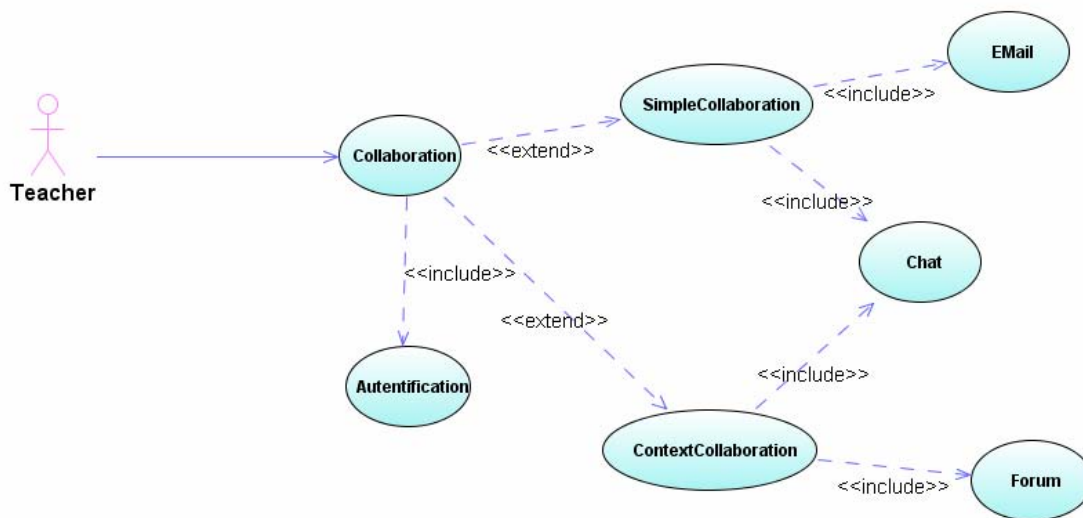


Ilustracija 61: Nastavničk aplikacija – SK 1. nivoa za upravljanje studentskim podacima

Ovaj SK ne uključuje administrativne funkcionalnosti dodavanja, brisanja, ili editovanja podataka studenata u sistemu. Postoji samo mogućnost pridruživanja studenata u kontekstualnu grupu, i ocenjivanje studenata u okvirima predmeta, čiji je nosilac nastavnik.

5.3.1.4 Saradnja sa drugim korisnicima

SK saradnje (*Collaboration*) odnosi se na različite vidove komunikacije između korisnika sistema (Ilustracija 63). Saradnja može da bude kontekstualna (*ContextCollaboration*) i uobičajena – vankontekstna (*SimpleCollaboration*). Kontekstualna saradnja znači da je sadržaj komunikacije striktno vezan za određeni kontekst učenja (na primer za određeni nastavni materijal, aktivnost, ili obavezu).



Ilustracija 62: Nastavnički aplikacija – SK 1. nivoa za saaradnju

Saradnja se odvija sinhronim (*chat*) i asinhronim (*e-mail*, forumi) vidovima komunikacije. Kontekstualna saradnja obuhvata forume i *chat*-ovanje, a vankontekstna *chat*-ovanje i *e-mail*. Kao u svim prethodnim slučajevima, autentifikacija korisnika je obavezan u svakom slučaju. Pored navedenih komunikacionih tipova, u sistem se mogu dodavati i novi (npr. video konferencijska komunikacija, govorna – *voip* komunikacija i td.). Predviđeno je da se svaki od tipova modularno razvija, a zatim uključuje u sistem.

5.3.2 Konceptualni modeli nastavničke aplikacije

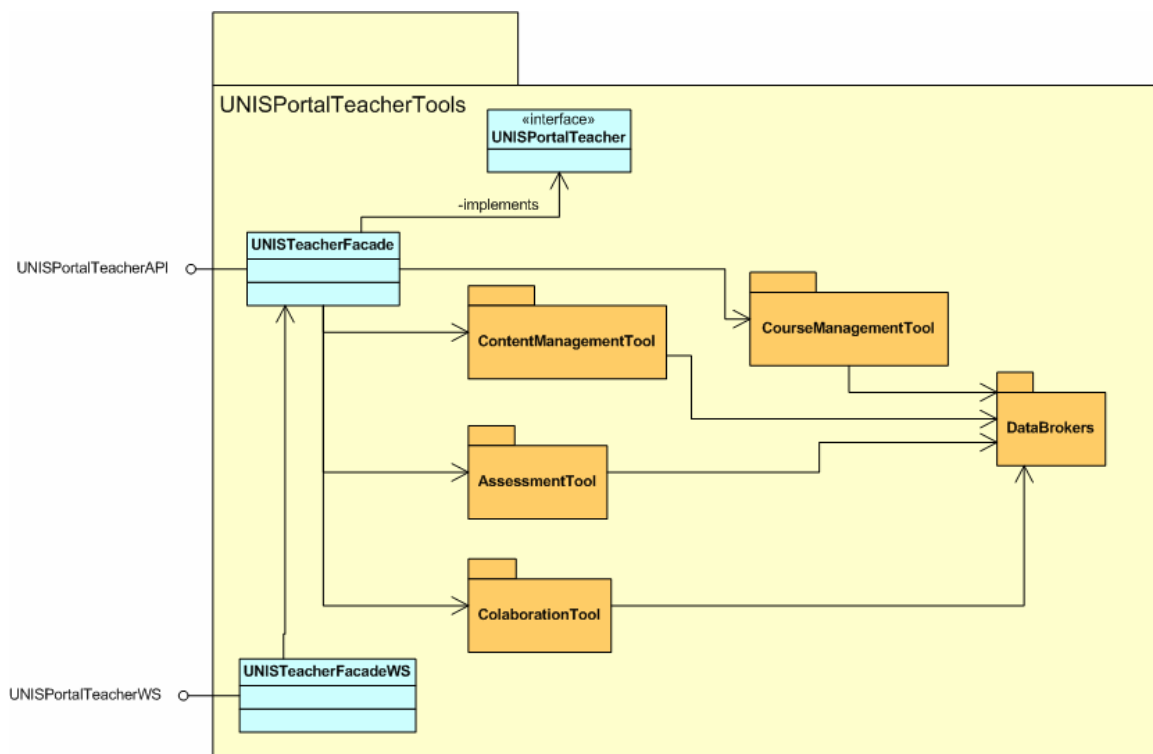
Klase nastavničke aplikacije su organizovane u dvonivojsku hijerarhiju paketa. Koreni paket je *UNISPortalTeacherTools*, koji sadrži pet pod-pakovanja (Ilustracija 64):

- *ContentManagementTool* – paket klasa za rad / upravljanje sadržajima učenja
- *CourseManagementTool* – paket klasa za rad / upravljanje predmetima
- *AssessmentTool* – paket klasa za rad / upravljanje proverama
- *CollaborationTool* – paket klasa za komunikaciju između korisnika
- *DataBrokers* – paket klasa – posrednika za rad sa podacima

Pored navedenih paketa u korenom paketu su definisani jedan interfejs (*UNISPortalTeacher*) i dve klase (*UNISTeacherFacade* i *UNISTeacherFacadeWS*). U interfejsu *UNISPortalTeacher* su deklarisanе metode posredstvom kojih korisnici (druge aplikacije) koriste klase nastavničkih alata.

Primenom *Facade* projektnog obrasca [113], dizajnirane su dve klase korenog paketa posredstvom kojih se komunicira izvana sa svim ostalim klasama paketa. Komunikacija se može odvijati na dva načina:

- Konvencionalno – posredstvom API (*ApplicationProgrammingInterface*-a), kada se predviđa ostvarivanje veće kompaktnosti sistema (npr. na istoj hardverskoj platformi sa korisničkim aplikacijama, i/ili implementacija u srodnim tehnologijama). U tom slučaju se koristi klasa *UNISTeacherFacade*.
- Posredstvom Web servisa – nekonvencionalno, kada se metode nastavničke aplikacije publikuju kao Web servisi, omogućavajući potpuno labavu spregu sa ostalim komponentama sistema. Na ovaj način moguća je sprega tehnološki potpuno različitih sistema (postizanje *napredne* interoperabilnosti na nivou poslovne logike). U opisu Web servisa nastavničke aplikacije pojavljuje se samo klasa *UNISTeacherFacadeWS*.



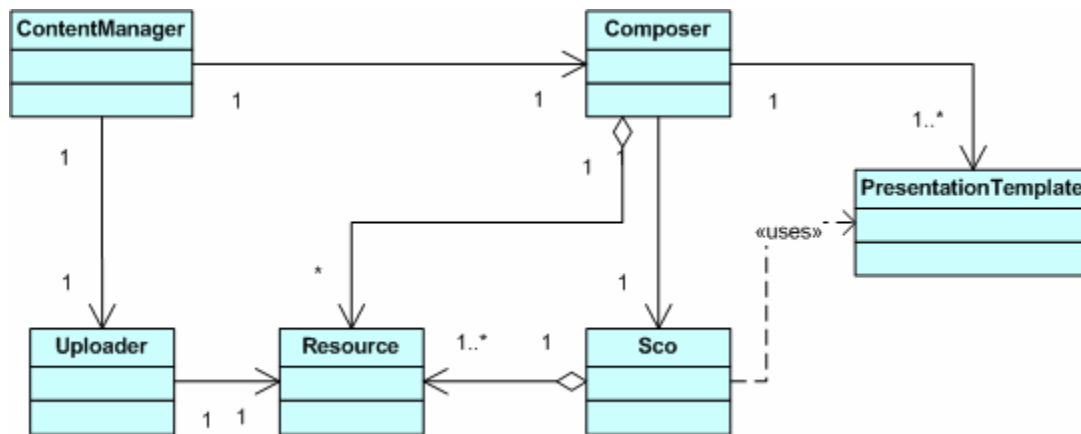
Ilustracija 63: Paketi klase nastavničke aplikacije

Klasa *UNISTeacherFacadeWS* ne napada direktno klase u pod-pakovanjima, već se obraća klasi *UNISTeacherFacade*, a ona dalje vrši propagaciju funkcionalnih poziva u dubinu sistema. Paket *DataBrokers* obuhvata klase koje su orjentisane prema podacima, dok sve ostale su orjentisane prema korisniku. *DataBrokers* praktično predstavlja zajednički resurs za ostale pakete u korenom - *UNISPortalTeacherTools*.

5.3.2.1 Paket *ContentManagmenetTool*

Paket *ContentManagmenetTool* namenjen je manipulaciji resursa učenja. Klasa *ContentManager* je posrednik između spoljnjih korisnika i ostalih klasa ovog paketa. Klasa *Uploader* je namenjena prihvatu od klijenta i postavljanju na sistem fajla resursa (SK *UploadContent*, poglavlje 5.3.1). Ova klasa prihvata fajl od serverske platforme (npr. *Weborb* platforma) posredstvom sistemskog objekta koji enkapsulira *upload*-ovan fajl.

U slučaju direktnog kreiranja resursa posredstvom aplikacije (SK *CreateContent*, poglavlje 5.3.1), koristi se klasa *Composer*. Ova klasa manipuliše deljivim resursima, uključujući ih u *SCO* objekat koji komponuje. U jednom trenutku korisnik može da komponuje jedan *SCO* objekat. *Composer* nudi korisnicima neki od predefinisanih prezentacionih šablona – instance klase *PresentationTemplate*. Odabrani šablon se referencira u *SCO* objektu (opisano stereotipnom zavisnošću *uses*).

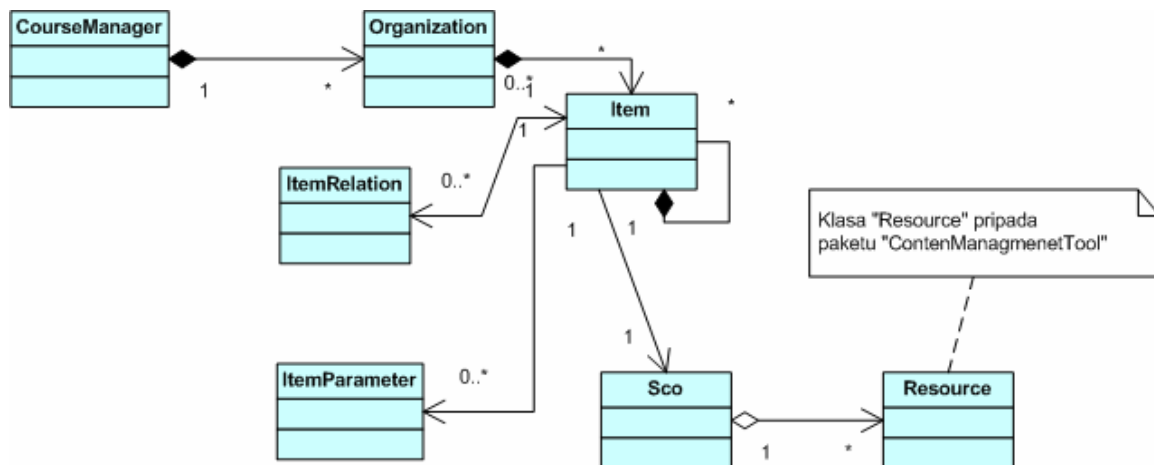


Ilustracija 64: Konceptualni model klasa paketa *ContentManagmenetTool*

Jedan *SCO* objekat predstavlja kompozit sastavljen od jednog, ili više objekata *Resource*. Klasa *Resource* enkapsulira osnovne podatke resursa (naziv, identifikator, ključne reči, tip, itd.) i referentnu putanju.

5.3.2.2 Paket *CourseManagmenetTool*

Ontologija predmeta je predstavljena paketom *CourseManagmenetTool*. Klasa *CourseManager* je posrednik između spoljnjih korisnika i ostalih klasa ovog paketa. Koncepti ontologije *Organization*, *Item*, *SCO*, *Resource*, preuzeti su iz SCORM 2004 standarda (poglavlje 3.2.4). Klasa *Organization* enkapsulira podatke predmeta, i sadrži instance klase *Item* (činilac). Činioci predmeta imaju dvojaku ulogu u modelu. Oni mogu da agregiraju pod-činioce, ili mogu da budu povezani na deljivi sadržaj učenja (*SCO*). *SCO* koncept omogućava da se grade složeni resursi učenja (npr. kombinacije slike i teksta, predstavljene u odabranom *template*-u). Oni zato predstavljaju kompozite, sastavljene od različitih fizičkih resursa. Svaki resurs je enkapsuliran klasom *Resource* (poglavlje 5.3.2.1).



Ilustracija 65: Konceptualni model klasa paketa *CourseManagementTool*

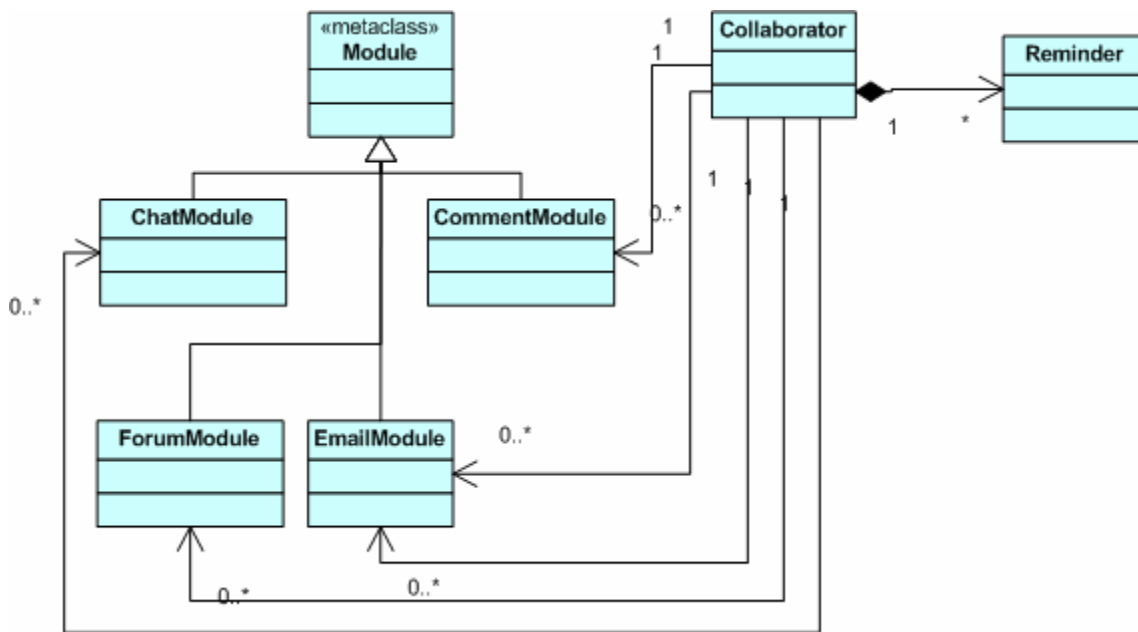
Pored hijerarhijskih veza, u ontologiji predmeta mogu postojati poprečne veze – veze između činilaca koji imaju zajedničkog roditelja (može da bude predmet, ili roditeljski činilac). Korišćenjem UML obrasca *Pure Fabricate* [113] napravljena je klasa *ItemRelation* koja čuva podatke o poprečnim vezama između činilaca. Ova klasa je vezana za klasu *Item* bidirekcionom asocijacijom, što znači da predstavlja sastavnicu između instanci *Item*-a.

Svaki činilac može da ima, pored regularnih i dodatne parametre, koji se čuvaju u nekom kontejneru (npr. vektor, ili *hash* tabela), kao instance klase *ItemParameter*. Koncept izdvojenih parametara omogućava da se doda neograničen broj dodatnih parametara i da se optimizira pretraga činilaca po dodatnim parametrima.

5.3.2.3 Paket *ColaborationTool*

Paket *ColaborationTool* sadrži klase namenjene komunikaciji između korisnika. Osnovne metode i atributi su deklarirani u apstraktnoj klasi *Module*, a za implementaciju su predviđene izvedene klase za svaki vid komunikacije: *ChatModule*, *ForumModule*, *EmailModule* i *CommentModule*.

Poslednja od navedenih klasa (*CommentModule*) nije namenjena za direktnu komunikaciju, već za personalizovanu anotaciju sadržaja. Ova klasa omogućava unos anotacija (komentara, zabeleški) za konkretan sadržaj. Ove anotacije mogu da budu privatne (vidljive samo autoru), ili javne (u slučaju da autor proglasi anotaciju vidljivom za ostale korisnike). To znači da javna anotacija ima kolaborativnu komponentu, jer omogućava korisnicima da sarađuju kroz komentare.

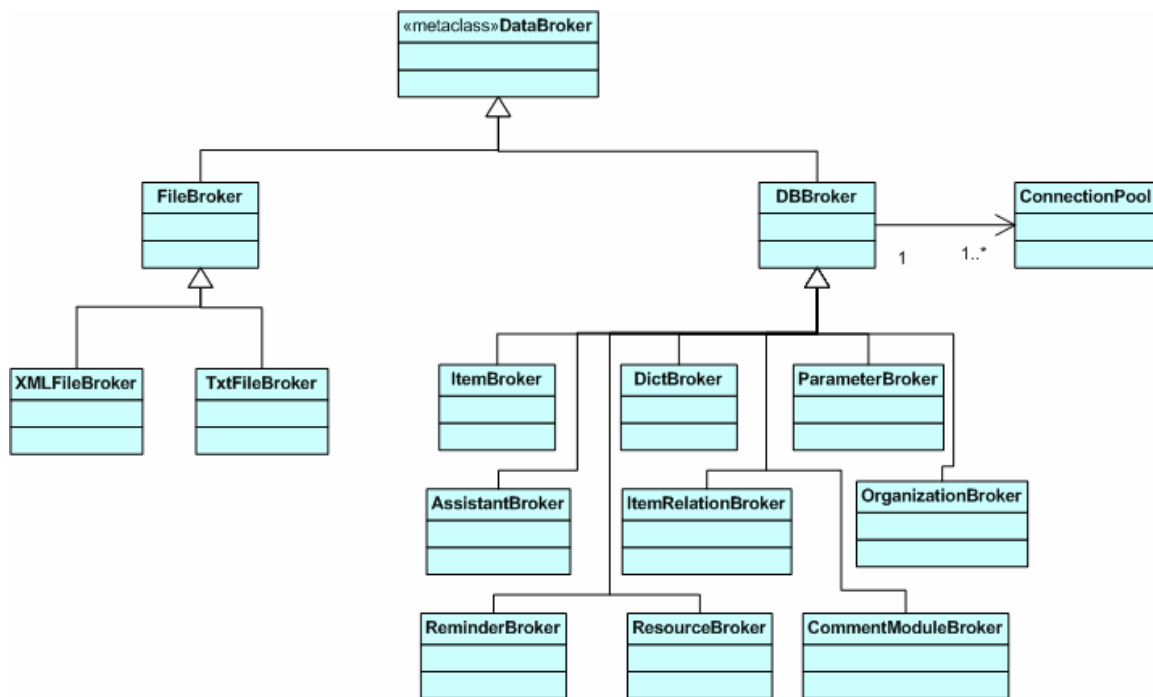


Ilustracija 66: Konceptualni model klasa paketa *ColaborationTool*

Korišćenje paketa *ColaborationTool* vrši se preko klase *Collaborator* (projektirani obrazac Facade). Pored navedenih, u paketu postoji još jedna klasa – *Reminder*. Ona je zadužena za prihvatanje podataka podsetnika. Svaki korisnik može da dodaje nove sadržaje podsetnika, ali je predviđeno da sistem koristi ovu klasu za automatsko generisanje i održavanje podsetnika, na osnovu definisanih aktivnosti i obaveza u budućnosti.

5.3.2.4 Paket *DataBrokers*

Svi prethodno opisani paketi predstavljaju konkretne alate za različite korisničke funkcionalnosti. Svi podaci i sve strukture, kreirane ovim alatima, postoje samo u radnoj memoriji aplikacije. Paket *DataBrokers* obuhvata klase koje omogućavaju serijalizaciju – deserijalizaciju svih podataka, merodavnih za nastavničku aplikaciju. Ovaj paket sadrži hijerarhijsku strukturu na čijem vrhu je apstraktna klasa *DataBroker* koja ima jedinu namenu da poveže dve raznorodne klasne strukture – za rad sa BP (*DBBroker*) i za rad sa fajl sistemom (*FileBroker*). Za rad sa resursima učenja projektovana je klasa *FileBroker*. Ova klasa da implementira primitive za rad sa fajlovima (kreiranje, preimenovanje, brisanje, premeštanje i zatvaranje fajlova). Sve ostale metode se definišu u izvedenim klasama (*XMLFileBroker* i *TxtFileBroker*). Ove klase su namenjene za manipulisanje podacima pri eksportovanju, ili importovanju predmeta, ili njegovih delova.



Ilustracija 67: Konceptualni model klasa paketa *DataBrokers*

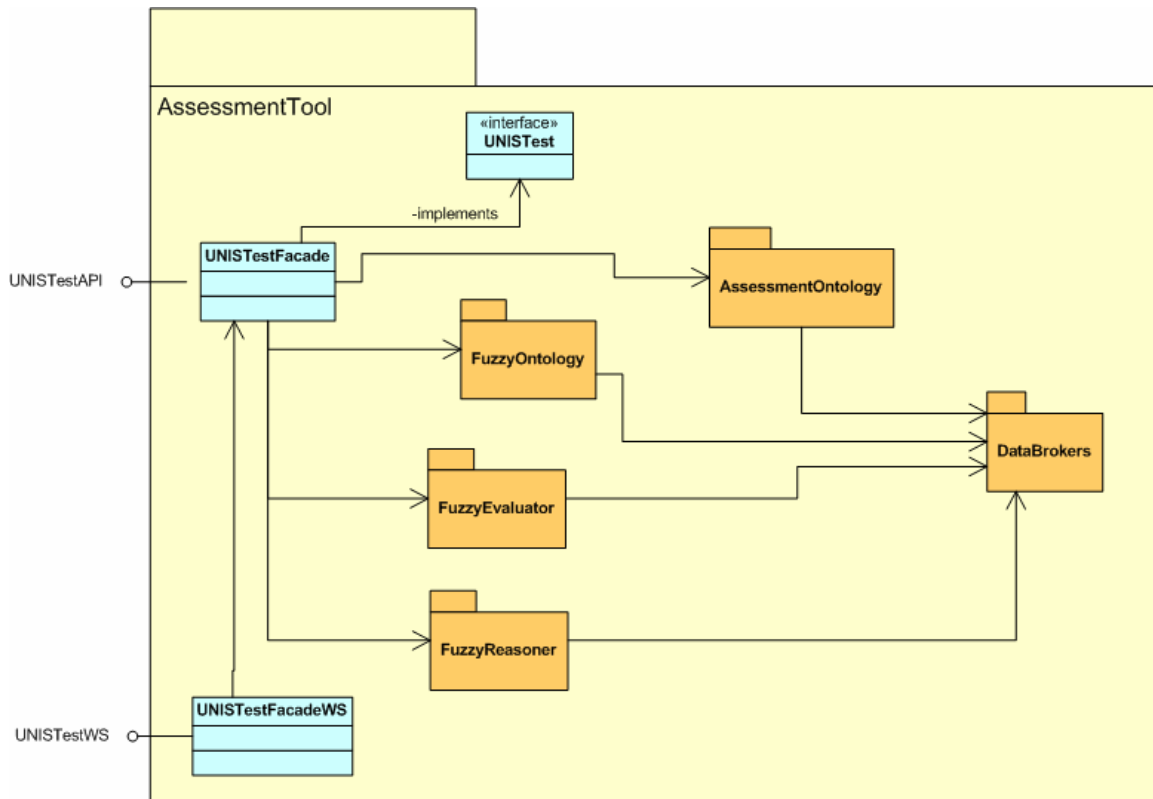
Osnovna zamisao je da se svi podaci čuvaju u BP. Klasa *DBBroker* implementira primitive za rad sa BP: uspostavljanje konekcije, davanje konekcije na korišćenje i prihvatanje korišćene konekcije. Za ove funkcionalnosti, koristi se klasa *ConnectionPool*. *DBBroker* može da sadrži više instanci *ConnectionPool*-a. Na taj način se rešava problem adaptiranja (UML projektni obrazac *Adaptor* [113]) na specifični sistem za upravljanje BP (npr. različiti *pool*-ovi za *MySQL*, *Oracle*, *Postgress*, itd.). Klasa *ConnectionPool* implementira i UML projektni obrazac *Object Pool* [67]. Ona čuva, uzima i vraća slobodne konekcije prema BP, tako da nije potrebno kreirati svaki put novu konekciju kada se zahteva izvršenje SQL naredbe (empirijski, više vremena se potroši na uspostavljanje konekcije, nego na izvršenje SQL naredbe).

Brokerske klase za rad sa konkretnim podacima (tabelama) su izvedene iz klase *DBBroker*. U ovim klasama su implementirane metode pregledanja, dodavanja, uklanjanja i menjanja podataka.

5.3.2.5 Paket *AssessmentTool*

Paket *AssessmentTool* je kompleksniji po strukturi od ostalih paketa, jer je polazna osnova za dizajn provera u sistemu je *IMS QTI* specifikacija za opis pitanja i testova (poglavlje 3.2.3 – *IMS Question & Test Interoperability*). Po ovoj specifikaciji, provere i pitanja (zadaci) predstavljaju specifičan slučaj (specijalizaciju) *Item*-a (činioca). Svi podaci jednog pitanja (tekst pitanja, ponuđeni odgovori, vrednosti (tačnost) odgovora, povratna poruka (*feedback*), namena pitanja, itd.) obuhvaćeni su jednim činiocem – *Item*-om (Ilustracija 69). Na primer, sadržaj pitanja obuhvaćen je *material* tag-om, ponuđeni odgovori *response* tag-ovima, ocenjivanje je definisano *resprocessing* tag-om, a povratna poruka (*feedback*) – *itemfeedback* tag-om.

Međutim, u *IMS QTI* specifikaciji postoji nekonzistentnost u odnosu na *IMS CP* specifikaciju iz koje je izvedena (poglavlje 3.2.3 – *IMS Content Packaging*). Preuzet koncept činilaca – *Item*-a u *IMS QTI* nije potpuno kompatibilan sa istoimenim, definisanim u *IMS CP* specifikaciji. Prema osnovnom opisu, postoje dva tipa činilaca: elementarni i struktuirani. Osnovni činioči nemaju pod-činioce i namenjeni su povezivanju za konkretan resurs. Struktuirani činioči imaju pod-činioce i ne mogu da se povežu za konkretan resurs. Resursi mogu da budu struktuirani (kao *SCO*), ili da predstavljaju jednu nedeljivu celinu koja sadrži, na primer, konkretan fajl sa opisom (*Assets*).

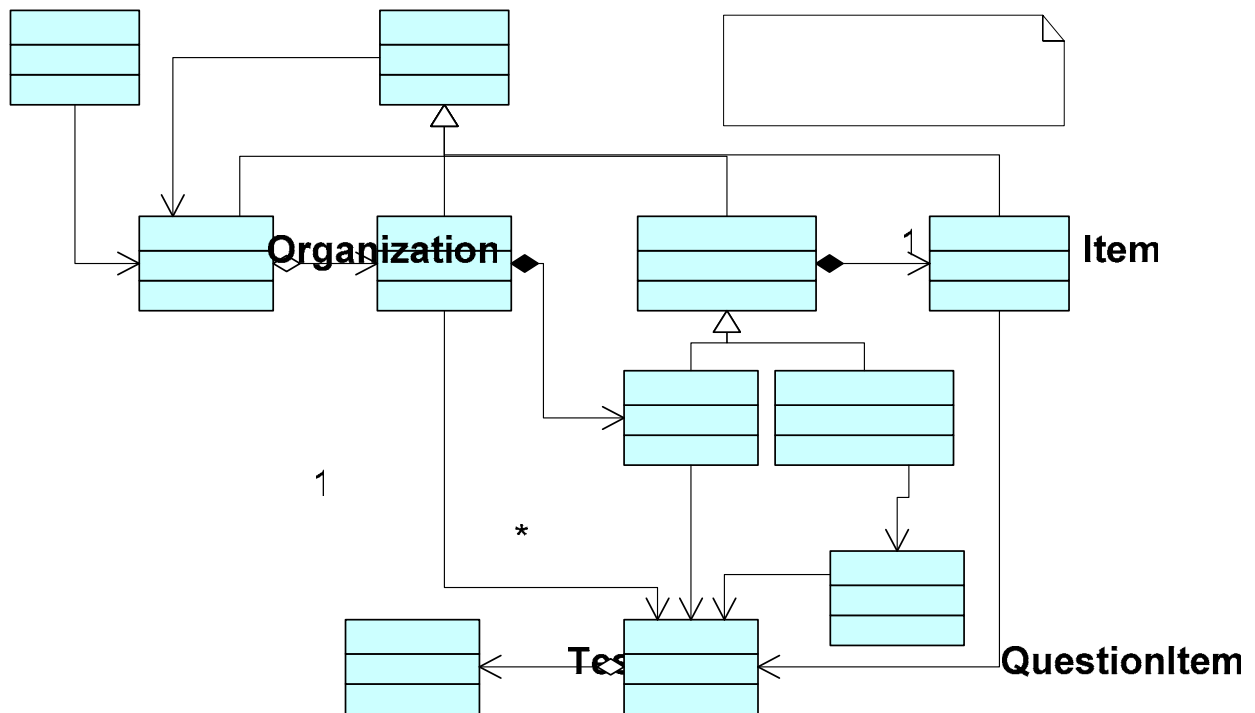


Ilustracija 68: Struktura nastavničkog alata za upravljanje proverama

U *IMS QTI* specifikaciji, pitanja su struktuirana i, između ostalih podataka, sadrže kolekciju materijala pitanja i odgovora. Materijali (tag *material*), takođe može biti kompozit (npr. kombinacija teksta i slika). Elementi materijala su praktično resursi (po *IMS CP* specifikaciji), koji treba da omoguće referenciranje van opsega definicije pitanja (npr. na fajlove slika, dokumenata i sl.).

Paket *AssessmentOntology*

Na osnovu ontologije organizacije predmeta, razvijena je posebna ontologija za provere (Ilustracija 70). Klase ove ontologije smeštene su u paketu *AssessmentOntology*



Ilustracija 69: Konceptualni model za definisanje provera

Ovaj konceptualni model ima kombinovanu strukturu koja se zasniva na izvođenju novih koncepata i njihovom povezivanjem sa postojećim konceptima. Iz klase *Item* izvedene su nove:

- *TestItem* – klasa koja enkapsulira kompletnu proveru. Testovi (provere) se definišu u okviru predmeta (*Organization*), ili činilaca predmeta (*Item*). Jedan predmet, ili činilac može da poseduje više provera. Jedna provera može da pripada samo jednom predmetu, ili činioocu. *TestItem* predstavlja kompozit više pitanja (objekti klase *QuestionItem*).
- *QuestionItem* – klasa koja enkapsulira jedno pitanje u okviru provere. Jedno pitanje može biti uključeno u različite provere. Pitanje sadrži jedan *SCO* objekat, posredstvom koga referencira na jedan ili više konkretnih resursa. Pored toga, pitanje poseduje i agregaciju odgovora.
- *GenericAnswer* – klasa koja u sebi sadrži sve što bi svaki odgovor trebao da ima, bez obzira na vrstu pitanja (izbor, slobodan unos, uparivanje i prevlačenje): identifikator, referenca na pitanje kome pripada, ocena i povratna informacija za korisnika. Odgovor poseduje i agregaciju povratnih informacija (*FeedbackItem*). Prema *IMS QTI* specifikaciji predviđeno je da odgovor ima jednu povratnu informaciju. Promena u agregaciju je napravljena da bi se omogućila adaptacija povratne informacije u zavisnosti od konteksta korišćenja (npr. samotestiranje u procesu učenja, ili testiranje za ispit).
- *AnswerItem* – klasa koja enkapsulira jedan odgovor u okviru pitanja tipa izbor (*xchoice*), ili slobodan unos (*fill in the blanks*). Odgovor može da pripada samo

jednom pitanju. Kao i pitanje, odgovor sadrži jedan *SCO* objekat, posredstvom koga referencira na jedan ili više konkretnih resursa.

- *PairedAnswerItem* – klasa koja enkapsulira jedan odgovor u okviru pitanja tipa uparivanje, ili prevlačenje (*connect, drag and drop*). Suštinski odgovori na ovaj tip pitanja se dobijaju uparivanjem, tako da svaki odgovor sadrži dva elementa tipa *PairItem* (još jedan nivo u hijerarhiji).
- *PairItem* – klasa koja pored identifikatora, reference na odgovor kome pripada, sadrži tip i vrednost sadržaja (tekst, ili naziv resursa)
- *FeedbackItem* – klasa koja enkapsulira povratnu informaciju povezanu za konkretan odgovor. Povratna informacija u sebi može da sadrži rešenje problema, objašnjenje, ili dodatnu pomoć. Iz tog razloga *feedback* referencira jedan objekat *SCO* i preko njega – jedan ili više konkretnih resursa.

Paket *DataBrokers* alata za ocenjivanje

U paketu alata za ocenjivanje postoji poseban pod-paket klasa namenjen serijalizaciji podataka. Ovaj paket je sličan istoimenom paketu klasa (poglavlje 5.3.2.4) roditeljskog pakovanja *UNISPortalTeacherTools*. Sa druge strane postoji specifičnost vezana za serijalizaciju fuzzy baze znanja. Ova posebna struktura, koja nema puno veze sa nastavnim materijalima i organizacijom nastave, sadrži uglavnom pedagoške aspekte ocenjivanja. Iz tog razloga *broker*-i ovog alata su izdvojeni u posebnom paketu.

5.4 Napredno ocenjivanje

Konvencionalni sistemi za upravljanje učenjem omogućavaju jednokriterijumsko ocenjivanje studenata. Najčešće se za test pitanja definišu bodovi, a rezultat studenata se predstavlja ukupnom broju sakupljenih bodova. Napredovanje kroz predmet (kurs) predstavlja akumulaciju bodova različitih provera. Često, rezultati izraženi u bodovima ne predstavljaju realnu sliku znanja studenata. Na primer, moguće su situacije da je u toku testiranja došlo do systemske greške, ili prekida konekcije između klijenta i servera. Studenti mogu da budu oštećeni od strane sistema. Adaptivno ocenjivanje u tom slučaju bi predstavljalo pokušaj sistema da na osnovu nepotpunih podataka oceni korisnika (na osnovu korisnikovog unosa do momenta greške).

Drugi aspekt naprednog ocenjivanja je prilagođavanje konkretnom sistemu ocenjivanja (*grading scales*). Skale ocenjivanja u različitim obrazovnim visokoškolskim sistemima se razlikuju (Tabela 10) [75]. To su skupovi unapred definisanih ocena koje se određuju bodovno, i/ili opisno, a u nekim sistemima nije definisano. U tom slučaju je prepušteno nastavnicima da sami određuju bodovne intervale, i/ili subjektivne kriterijume.

Isti elementi na skalama različitih sistema nose različitu semantiku. Na primer u BiH i u Sloveniji, koristi se identična skala kao u našem visokoškolskom obrazovanju (skala od 5 do 10). Međutim u BiH ocena 9 predstavlja *vrlo dobru* ocenu, a ocene 7 i 8 su *dobre* ocene. Kod Slovenačkog sistema ocene 8 i 9 su *vrlo dobre* ocene. U nekim okolnim zemljama koristi se na različite načine skala od 1 do 5 (npr. Hrvatska i Slovačka). U Hrvatskoj 5 je najbolja ocena, a 1 najgora; u Slovačkoj je potpuno obrnuto.

Tabela 10: Skale za ocenjivanje u različitim obrazovnim sistemima

Sistem ocenjivanja	Skala	Opis	Opsezi
ECTS (European Credit Transfer and Accumulation System)	A	Excellent	90%–100%
	B	Very good	65%–90%
	C	Good	35%–65%
	D	Satisfactory	10%–35%
	E	Sufficient	0%–10%
	FX	Fail, more work required	-
	F	Fail, considerably more work required	-
SAD	A	excellent	100-90
	B	above average	89-80
	C	average	79-70
	D	usually the minimum passing grade	69-60
	F	fail	Ispod 60
Australija	Različiti sistemi u različitim državama		
Novi Zeland National Certificate of Educational Achievement (NCEA)	E	Excellence	Nije precizirano (samo opisi ocena)
	M	Merit	
	A	Achieved	
	N ili NA	Not achieved	
Japan	A	-	90%–100%
	B	-	70%–89%
	C	-	60%–69%
	F	-	0%–59%
Bosna i Hercegovina	10	odličan	-
	9	vrlo dobar	-
	7-8	dobar	-
	6	dovoljan	-
	5	nedovoljan	-
Hrvatska	5	odličan	-
	4	vrlo dobar	-
	3	dobar	-
	2	dovoljan	-
	1	nedovoljan	-
Bugarska	6	odličan	-
	5	vrlo dobar	-
	4	dobar	-
	3	dovoljan	-
	2	nedovoljan	-

U najvećem broju obrazovnih sistema, postoje slovčane (alfabetske) skale od A do D (npr. u Estoniji). Pored navedenih razlika u sistemima za ocenjivanje, postoje i razlike u semantici pojedinih ocena. Na primer, kod skale 1 – 5 u nekim sistemima petica ima semantiku najviše ocene, ali u drugim ima semantiku najmanje ocene.

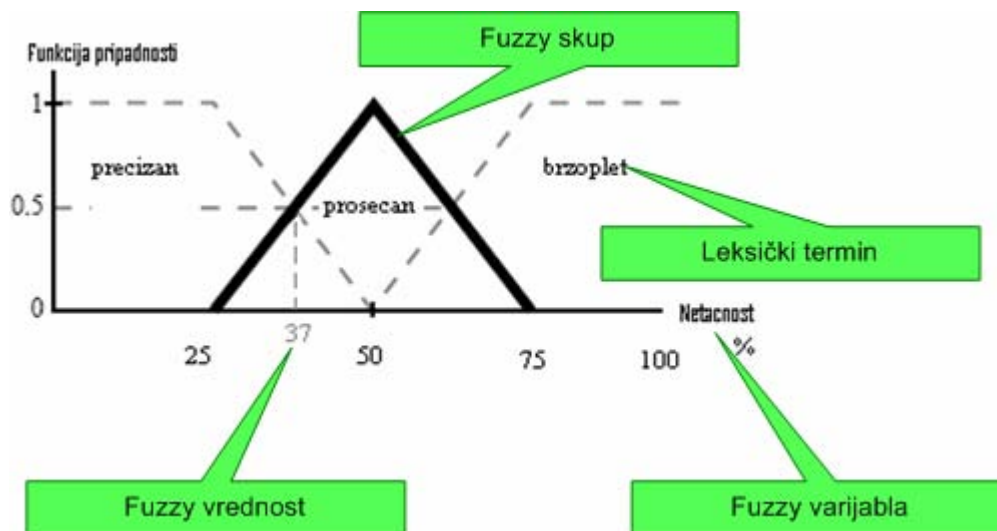
Nije neuobičajeno da se na ocene dodaju i modifikatori (na primer *, +, -), koji treba preciznije da opišu osnovnu ocenu (u Mađarskoj). U obrazovnim sistemima Švajcarske i Nemačke taj problem je rešen proširivanjem skale (od najbolje 6, do najgore ocene 1), jer je za mašinsko procesiranje rezultata problematično uvođenje specifičnih znakova koji nisu cifre. Predloženo rešenje omogućava adaptaciju ocenjivanja prema zahtevima i uslovima konkretnog edukativnog sistema.

5.4.3 Korišćenje fuzzy logike u sistemu

Fuzzy logika je grana veštačke inteligencije namenjena mašinskom rezonovanju na osnovu nepreciznih (približnih) podataka [68]. Fuzzy logika može da se razmatra kao proširenje konvencionalne logike (propozicione i predikatske logike). Osnovni logički postulat je da činjenice i iskazi mogu da budu tačni, *ili* netačni. Fuzzy logika dozvoljava da činjenice i iskazi mogu da budu tačni *i* netačni, zapravo *delimično* tačni *i* *delimično* netačni. Ovaj paradoks je rešen u fuzzy logici konceptom funkcije pripadanja. Za svaku činjenicu, ili iskaz, tačnost se izražava vrednošću funkcije pripadanja razmatranom skupu.

5.4.3.1 Korišćeni koncepti

U realnom svetu, česta je potreba da se donose odluke, ili zaključci na osnovu neegzaktnosti. Ova grana logike omogućava da se u mašinama emulira način ljudskog razmišljanja i donošenja odluka. Da bi se omogućila praktična implementacija, definišu se fuzzy koncepti: fuzzy skup, fuzzy varijabla, fuzzy termin, fuzzy pravilo i fuzzy vrednost.



Ilustracija 70: Fuzzy koncepti korišćeni u sistemu

Da bi se fuzzy logika koristila za zaključivanje o nekom svojstvu iz realnog sveta, najpre se definiše varijabla za to svojstvo. U primeru sa slike (Ilustracija 71), svojstvo o kome se razmatra je *netačnost*. Definisanje varijable obuhvata dodelu imena varijabli, određivanje skupa vrednosti koji će se razmatrati i definisanje jedinca mere za izražavanje konkretnih vrednosti svojstava. Varijabla iz primera je *netačnost* (svojstvo koje se razmatra je netačnost studenata ispoljena u toku testova). U praksi je neophodno da se fuzzy varijabla predstavi konačnim skupom vrednosti - *univerzumom* (*universe of discourse*). U primeru koji se obrađuje, ovaj zahtev nije problematičan, jer se vrednost po fuzzy varijabli izražava procentualno [0 – 100], ali postoje slučajevi kada je skup mogućih vrednosti neograničen (npr. za rastojanja, temperaturu, pritisak, brzinu i sl.).

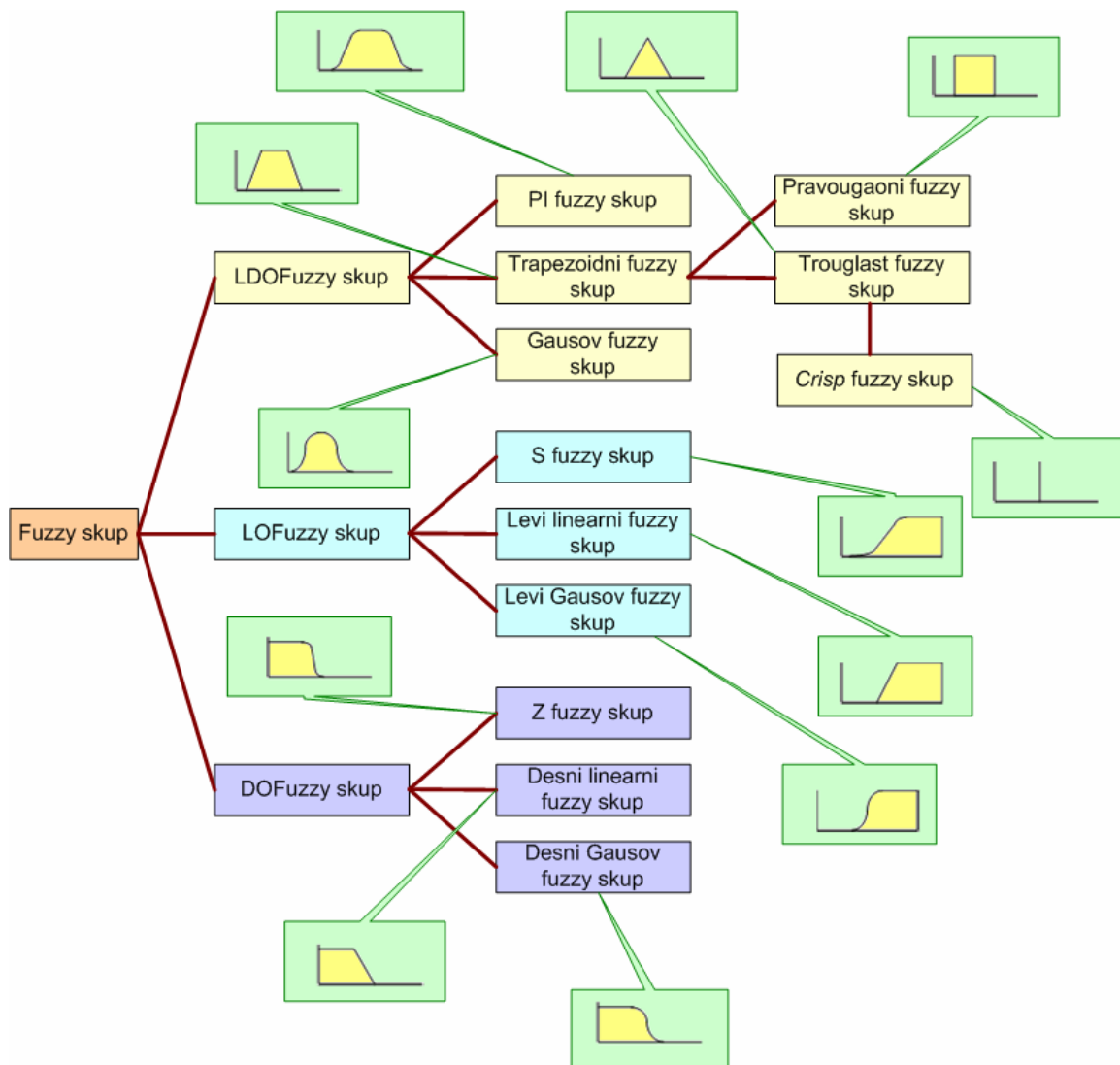
Fuzzy skupovi predstavljaju pod-skupove univerzuma fuzzy varijable. Elementi fuzzy skupova su uređeni parovi konkretnih vrednosti fuzzy varijable i funkcije pripadanja te vrednosti razmatranom fuzzy skupu. Funkcija pripadanja (*grade of membership*) ima

normalizovane vrednosti u intervalu $[0 - 1]$. Ovim je omogućeno da se rezonovanje Boolean-skomp logikom $\{0, 1\}$ zameni fuzzy rezonovanjem $[0 - 1]$. Za razliku od konvencionalnih skupova, kod kojih vrednosti za posmatranu varijablu (svojstvo) ili pripada, ili ne pripada skupu, kod fuzzy skupova svaka vrednosti pripada/ne pripada skupu u određenoj meri – izraženoj kao vrednosti funkcije pripadanja. Potpuno pripadanje/ne pripadanje predstavljaju specijalne slučajeve, kada su vrednosti funkcije pripadanja za posmatrano svojstvo 1, odnosno 0.

Fuzzy skup (kao i fuzzy varijabla) može da bude imenovan leksičkim terminom. Terminima se jezički opisuje svojstvo. Na prethodnom primeru, svojstvo *netačnost* opisano je sa tri leksička termina: *precizan*, *prosečan* i *brzoplet*. Iz aspekta fuzzy logike, to su tri imenovana fuzzy skupa: Z skup (*precizan*) $[0 - 50\%]$, trouglast skup (*prosečan*) $[25 - 75\%]$ i S skup (*brzoplet*) $[50 - 100\%]$.

5.4.3.2 Tipovi fuzzy skupova

Tipologizacija fuzzy skupova vrši se na osnovu njihove grafičke reprezentacije (Ilustracija 72). Svi fuzzy skupovi se klasifikuju u tri velike grupe: ograničeni s leve strane – LOFuzzy skupovi (eng.*LFuzzySet*), ograničeni s desne strane – DOFuzzy skupovi (eng.*RFuzzySet*), i ograničeni i sa leve i sa desne stran – LDOFuzzy skupovi (eng.*LRFuzzySet*).



Ilustracija 71: Tipovi fuzzy skupova

Fuzzy skupovi ograničeni s leve strane

Postoje tri tipa fuzzy skupova ograničenih s leve strane:

- S fuzzy skup – skup čija se 0 (nula vrednost) funkcije pripadanja nalazi s leve strane, a 1 funkcije pripadanja nalazi na desnoj strani dijagrama. Dijagram je kompozit fragmenata dve linearne funkcije $y=ax+b$ ($a>0$), $y=1$, i fragmenata dve nelinearne funkcije (npr. dve kvadratne funkcije, ili eksponencijalne i logaritamske funkcije).
- Levi linearni fuzzy skup – skup čija se 0 (nula vrednost) funkcije pripadanja nalazi s leve strane, a 1 funkcije pripadanja nalazi na desnoj strani dijagrama. Dijagram je kompozit fragmenata dve linearne funkcije: $y=ax+b$ ($a>0$), i $y=1$.
- Levi Gausov fuzzy skup – skup čija se 0 (nula vrednost) funkcije pripadanja nalazi s leve strane, a 1 funkcije pripadanja nalazi na desnoj strani dijagrama.

Dijagram je kompozit fragmenata dve funkcije: Gausove (Izraz 2) i linearne funkcije $y=1$

$$y = e^{-\frac{(x-a)^2}{2\delta^2}}, \quad \begin{array}{l} a - \text{srednja vrednost,} \\ \delta - \text{standardna devijacija} \end{array} \quad (2)$$

Fuzzy skupovi ograničeni s desne strane

Analogno fuzzy skupovima ograničenim s leve strane, postoje tri tipa fuzzy skupova ograničenih s desne strane:

- Z fuzzy skup – skup čija se 0 (nula vrednost) funkcije pripadanja nalazi s desne strane, a 1 funkcije pripadanja nalazi na levoj strani dijagrama. Dijagram je kompozit fragmenata dve linearne funkcije: $y=ax+b$ ($a<0$) i $y=1$, i dve nelinearne funkcije (npr. dve kvadratne funkcije, ili eksponencijalne i logaritamske funkcije).
- Desni linearni fuzzy skup – skup čija se 0 (nula vrednost) funkcije pripadanja nalazi s desne strane, a 1 funkcije pripadanja nalazi na levoj strani dijagrama. Dijagram je kompozit fragmenata dve linearne funkcije: $y=ax+b$ ($a<0$) i $y=1$.
- Desni Gausov fuzzy skup – skup čija se 0 (nula vrednost) funkcije pripadanja nalazi s desne strane, a 1 funkcije pripadanja nalazi na levoj strani dijagrama. Dijagram je kompozit fragmenata dve funkcije: Gausove (Izraz 2) i linearne funkcije $y=1$

Fuzzy skupovi ograničeni i s leve i s desne strane

Fuzzy skupovi ograničeni s i leve i s desne strane su tipski kompleksniji. Postoje tri osnovna tipa:

- Gausov fuzzy skup – skup kod koga se funkcija pripadnosti menja isključivo po Gausovoj funkciji – normalna raspodela vrednosti (Izraz 2).
- PI fuzzy skup – predstavlja kombinaciju S i Z fuzzy skupa. Vizuelno je sličan Gausovom fuzzy skupu, s tim da ima segment linearne funkcije $y=1$, i zakrivljenost leve i desne strmine nije po Gausovoj funkciji, već predstavlja kompozit fragmenata dve linearne funkcije: $y=ax+b$ (za negativnu strminu $a<0$, za pozitivnu strminu $a>0$), i četiri fragmenta dve nelinearne funkcije (npr. dve kvadratne funkcije, ili eksponencijalne i logaritamske funkcije).
- Trapezoidni fuzzy skup – predstavlja kombinaciju levog i desnog linearnog fuzzy skupa. Dijagram skupa je trapezoid – kompozit tri linearne funkcije: $y=a_1x+b_1$ ($a_1>0$) za pozitivnu strminu, $y=a_2x+b_2$ ($a_2<0$) za negativnu strminu, i segment temena $y=1$.

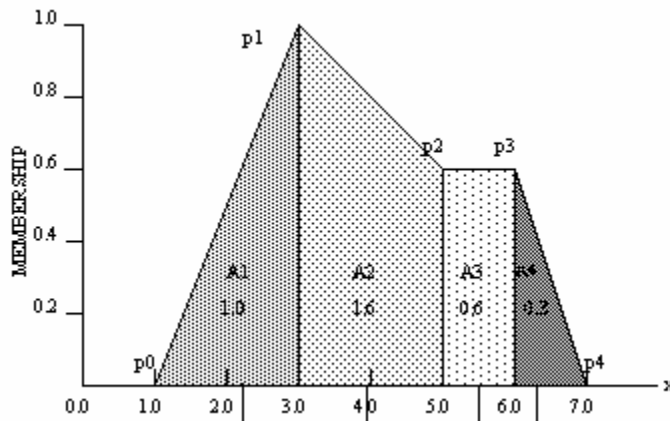
Trapezoidni fuzzy skup se deli dalje na pod-tipove:

- Pravougaoni fuzzy skup – predstavlja se jednim linearnim segmentom $y=1$ u ograničenom opsegu ulazne varijable (kao uniformna raspodela).
- Trouglasti fuzzy skup – predstavlja specifičan slučaj trapezoidnog fuzzy skupa, kad ne postoji linearni segment $y=1$. Dijagram skupa je trougao - kompozit dve

linearne funkcije: $y=a_1x+b_1$ ($a_1>0$) za pozitivnu strminu, $y=a_2x+b_2$ ($a_2<0$) za negativnu strminu.

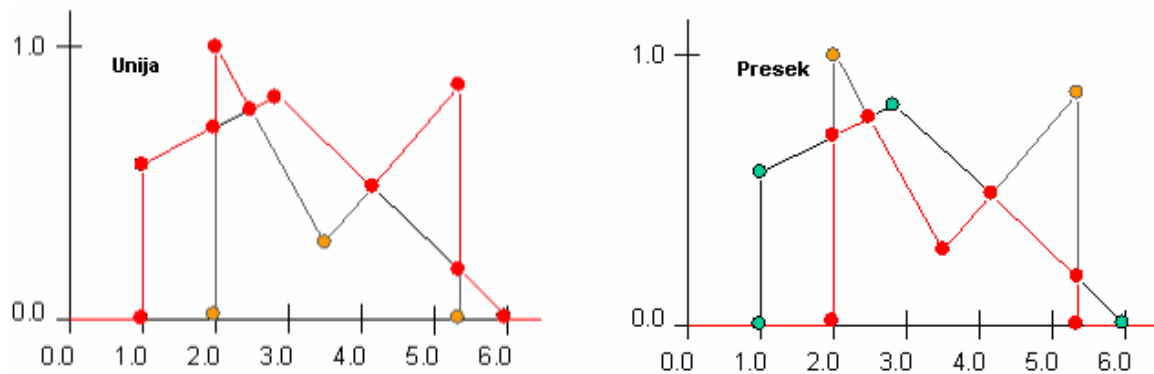
- *Crisp* fuzzy skup – najspecifičniji fuzzy skup, izveden iz trouglastog (poklapanjem pozitivne i negativne strmine, koje su pri tome vertikalne), kojim se u fuzzy logici predstavlja diskretna vrednost. To je skup od samo jedne vrednosti ulazne promenljive ($y=\{Kx_1\}$).

Pored navedenih osnovnih tipova, fuzzy skupovi mogu biti i izvedeni kao njihovi kompoziti (Ilustracija 73).



Ilustracija 72: Primer kompozitnog fuzzy skupa

Na prethodnom dijagramu je primer kompozitnog fuzzy skupa, sastavljenog od četiri skupa osnovnog tipa: dva trouglasta (A1 i A4), jednog trapezoidnog (A2) i jednog pravougaonog (A3). Fuzzy skupovi mogu da se dobiju i na osnovu operacija negacije, preseka i unije nad drugim fuzzy skupovima (Ilustracija 74).



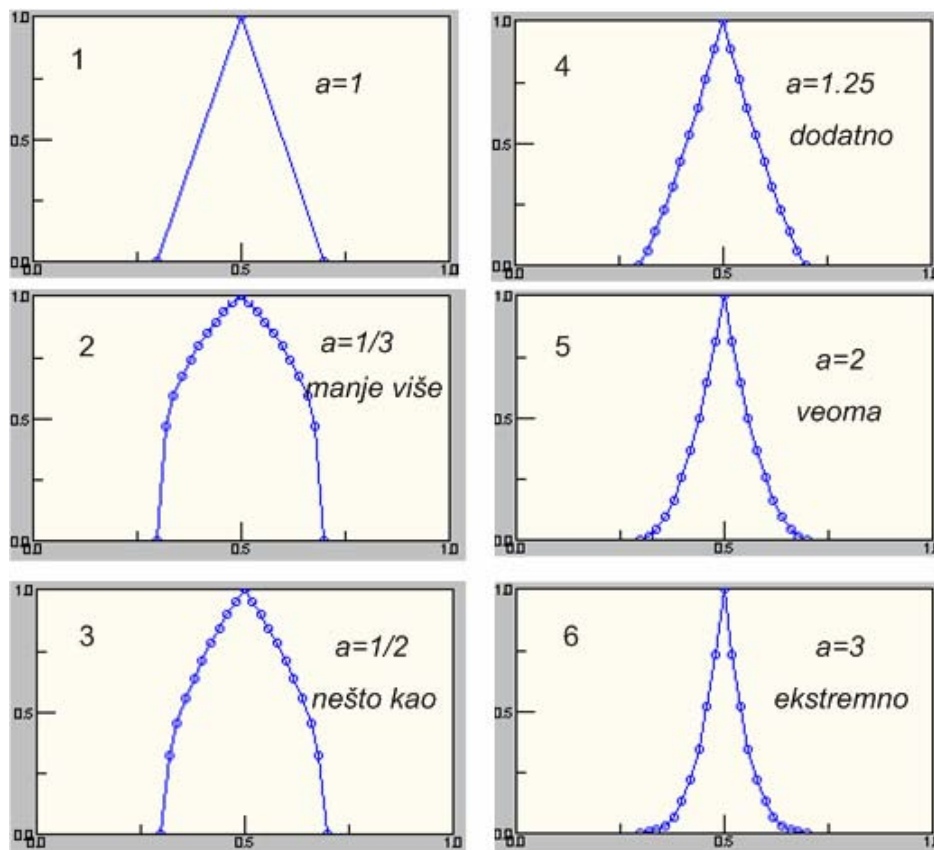
Ilustracija 73: Izvođenje fuzzy skupa korišćenjem operacija unije i preseka

Zbog otežane predikcije rezultata fuzzy rezonovanja, ulazne vrednosti fuzzy pravila najčešće se predstavljaju osnovnim tipovima fuzzy skupova. Nepravilni fuzzy skupovi (kompoziti dobijeni presecima ili unijama) se najčešće dobijaju kao rezultat procesa rezonovanja (poglavlje 5.4.7.5).

Modifikatori

Fuzzy skupovi mogu da se dobiju i modifikacijom osnovnih tipova pomoću modifikatora. Modifikatori se koriste radi preciznijeg definisanja svojstava nad kojima se vrši fuzzy rezonovanje. Svi prethodno opisani fuzzy skupovi su nemodifikovani.

Modifikatori su naročito korisni ako se u fuzzy pravilima (znači – za rezonovanje) koriste leksički termini. Korišćenjem modifikatora se proširuje osnovni skup korišćenih termina, dodavanjem priloga za način (vrlo, veoma, neznatno, donekle, ekstremno, i sl.). Modifikacije se najčešće primenjuju nad linearnim fuzzy skupovima, stepenovanjem funkcije pripadanja odgovarajućim koeficijentom ($y_{mod}=y^a$).



Ilustracija 74: Modifikovan trouglasti fuzzy skup

Na prethodnoj ilustraciji (Ilustracija 75) predstavljeni su karakteristični primeri upotrebe modifikatora. Nemodifikovan fuzzy skup (koeficijent modifikacije $a=1$) za neko svojstvo nad kojim se rezonuje predstavljen je na dijagramu br.1; Za vrednosti koeficijenta manjim od jedan, dobijaju se modifikovani fuzzy skupovi koji su u vrhu *zaobljeni*, i koji se leksički izražavaju priložima *manje-više* (dgr.br.2) i *nešto kao* (dgr.br.3). Za koeficijent modifikacije veći od jedan, zaoštavaju se kriterijumi pripadnosti fuzzy skupu. Prilozi koji se koriste za jezičko izražavanje su *dodatno* (dgr.br.4), *veoma* (dgr.br.5) i *ekstremno* (dgr.br.6).

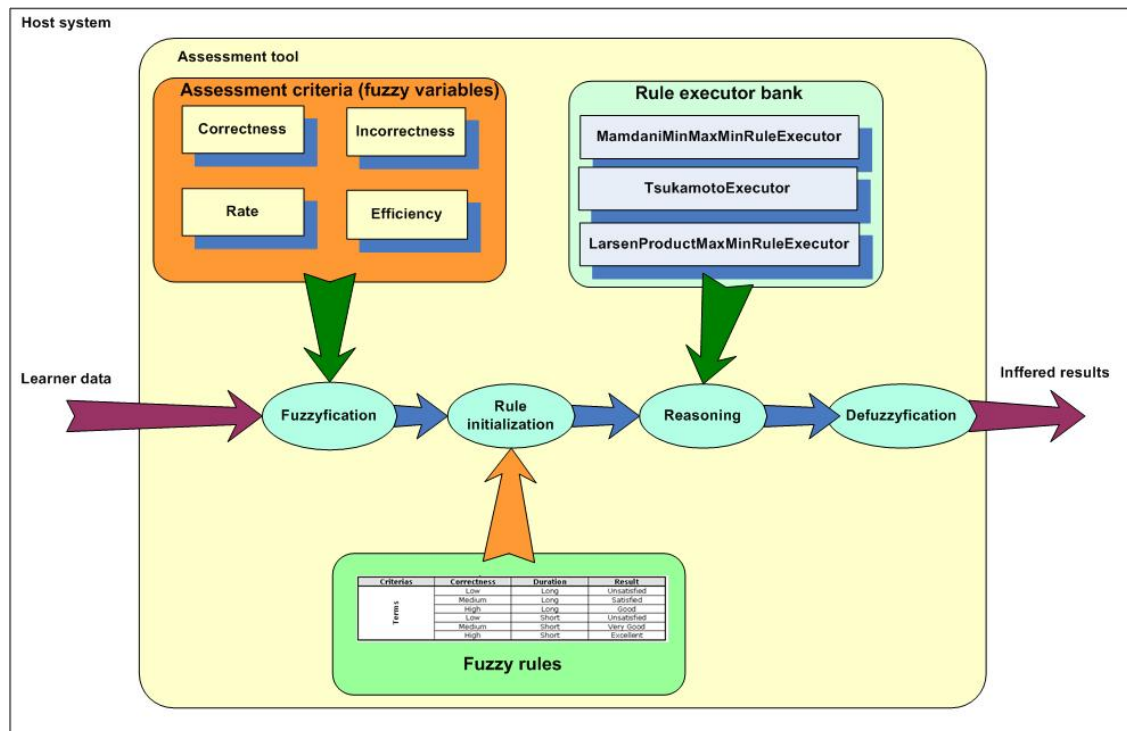
Pored navedenih postoje i drugi modifikatori, koji se koriste nad različitim fuzzy skupovima i koji smanjuj ili pojačavaju pripadnosti vrednosti svojstva fuzzy skupu. Takođe, često je korišćenje modifikatora u kombinaciji sa negacijom da bi se izrazila specifična svojstva (npr.*nije ekstremno*, ili *nije kao*). Specifičan slučaj modifikacije je

normalizacija fuzzy skupa. Pri operacijama sa skupovima, a naročito kod preseka skupova, vrednosti funkcije pripadanja rezultujućeg fuzzy skup su mnogo manje od 1. Time se njegov uticaj na rezonovanje slabo ispoljava. Normalizacijom (dovođenjem vrednosti funkcije pripadanja u ceo interval $[0 - 1]$) se efekti operacija nad fuzzy skupovima kompenzuju.

Da bi fuzzy sistem mogao da rezonuje pouzdano i bez grešaka, fuzzy skupovi moraju da budu definisani nad celokupnim univerzumom varijable. To znači da svaka vrednosti univerzuma varijable pripada barem jednom fuzzy skupu, i ne postoji vrednosti univerzuma varijable koja ne pripada ni jednom fuzzy skupu (slično konceptima univerzalnog i egzistencijalnog kvantifikatora predikatske logike).

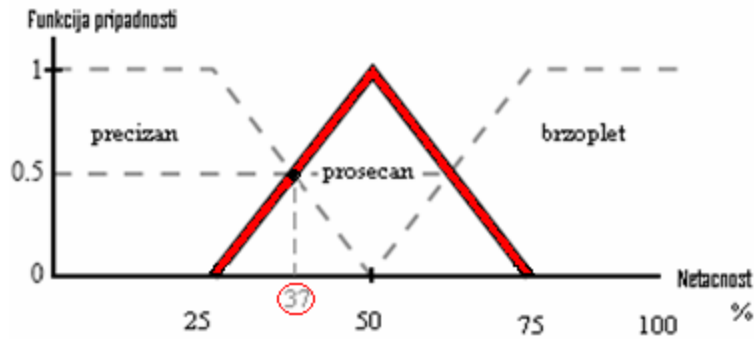
5.4.3.3 Fazifikacija

Fuzzy rezonovanje korišćeno u sistemu odvija se kroz četiri faze (Ilustracija 76): fazifikacija, inicijalizacija fuzzy pravila, rezonovanje i defazifikacija.



Ilustracija 75: Faze fuzzy rezonovanja

Proces transformacije diskretnih ulaznih podataka u fuzzy vrednosti naziva se *fazifikacija*. Ulazni podaci su diskretni, tako da treba da se prilagode fuzzy egzekutoru. U fazifikaciji se generiše fuzzy vrednosti od ulaznih podataka korišćenjem odgovarajuće fuzzy varijable i fuzzy skupa (fuzzy ontologija, Ilustracija 77). Praktično se računa funkcija pripadanja ulazne vrednosti u pridruženom fuzzy skupu. U slučaju da ne postoji definisana fuzzy varijabla za diskretnu ulaznu vrednost, ista se ne uključuje u fuzzy rezonovanje.



Ilustracija 76: Primer fazifikacije diskretne vrednosti izražene u procentima

Na primeru (Ilustracija 77), netačnost studenta na proveru od 37% je diskretna ulazna vrednost. Ona se fazifikacijom transformiše u fuzzy vrednost, koja se može leksički opisati kao: „student X je *polovično* (0.5) *precizan*, *polovično* (0.5) *prosečan* i *nije* (0) *brzoplet*“. To znači da se za istu vrednost računaju funkcije pripadanja u svim fuzzy skupovima, predefinisanim nad univerzumom fuzzy varijable.

5.4.3.4 Fuzzy pravila i njihova inicijalizacija

Fuzzy pravila su generalno slična pravilima u diskretnoj (bivalentnoj) logici. Ona sadrže uslovni deo - *premisu* i akcioni deo - *konsekvencu*. U fuzzy pravilima se vrši preslikavanje fuzzy skupova vrednosti u premisi u fuzzy skupove vrednosti koje se nalaze u zaključku pravila. U fuzzy rezonovanju najčešće korišćen tip pravila su *Mamdani* fuzzy pravila [69]. Ova pravila rade na bazi *modus ponens*-a.

$$x_1 = A_1 \wedge x_2 = A_2 \dots x_i = A_i \rightarrow y_1 = C_1, y_2 = C_2 \dots y_j = C_j \quad (3)$$

Mamdani pravila imaju dva dela – premisu i zaključak. Oba dela mogu biti složena. U premisi se nalazi konjunkcija (ulaznih) vrednosti, predstavljenih fuzzy skupovima. Zaključak *Mamdani* pravila je skup (izlaznih) vrednosti, takođe predstavljenih fuzzy skupovima. U predstavljenom izrazu (Izraz 3) postoji *i* ulaznih vrednosti *x*, svaka od njih je predstavljena fuzzy skupom *A*. Predstavljeno je *j* izlaznih vrednosti *y*, predstavljenih fuzzy skupovima *C*.

Fuzzy pravila moraju da budu u potpunosti definisana pre procesa rezonovanja. To znači da fuzzy skupovi u premisi [*A*₁, *A*_{*i*}] i zaključku [*C*₁, *C*_{*j*}] moraju da budu uključeni u pravila, tako da su ona spremna da se inicijalizuju ulaznim (fazifikovanim) vrednostima [*x*₁, *x*_{*i*}]. Fuzzy pravila u sistemu su iskorišćena za modelovanje nastavničkog iskustva u ocenjivanju studenata (poglavlje 5.4.7).

Pored *Mamdani* pravila, postoje i drugi tipovi. *Takagi-Sugeno-Kang* (*TSK*, ili kraće *TS*) pravila [132] imaju kao ulaze fuzzy vrednosti, ali za razliku od *Mamdani* pravila, rezultati rezonovanja su diskretne vrednosti. Kod *TS* pravila postoje dve vrste izlaznih vrednosti: konstante (tzv. *TSK* pravila *nultog reda*), ili varijable koje su definisane kao funkcije za izračunavanje diskretne vrednosti (tzv. *TS* pravila *prvog reda*). Najčešće korišćena funkcija ima 2 ulazna podatka i definiše se kao u sledećem izrazu (Izraz 4).

$$z = ax + by + c \quad (4)$$

TSK pravila *nultog reda* su specijalni slučajevi kada su koeficijenti jednaki nuli ($a=0$, $b=0$).

Tsukamoto pravila [133] su veoma slična *TSK* pravilima. Njihova specifičnost su izlazne funkcije, koje moraju da budu striktno rastuće, ili opadajuće. Na primer, to su funkcije koje se mogu grafički predstaviti levo, ili desno ograničenim fuzzy skupovima (poglavlje 5.4.3.1). Kada se izvrši *Tsukamoto* pravilo, na izlazu se dobija fuzzy vrednost u vidu *crisp* fuzzy skupa (diskretna vrednost i njena funkcija pripadanja). Smisao funkcije pripadanja je da izrazi stepen poklapanja/ispunjenja pravila (*DOF* – *Degree of Fulfillment*). Izlazna fuzzy vrednost se bira iz zaključnog fuzzy skupa na osnovu poklapanja funkcije pripadanja.

5.4.3.5 Fuzzy rezonovanje

Sušтина fuzzy rezonovanja je da se sve ulazne vrednosti nakon fazifikacije primene na sva pravila u fuzzy bazi znanja, a zatim da se izvrše *odgovarajuća* fuzzy pravila (u kojima postoji prihvatljivo poklapanje premisa). Pravila se izvršavaju sekvencijalno (u jednom trenutku – samo jedno pravilo). Za svako pravilo, sistem treba da doda njegovu izlaznu vrednost (rezultat) u izlazni skup. Na taj način se vrši akumulacija rezultata na globalnom nivou.

Postoje različite metode rezonovanja. Najpoznatije metode su *Mamdani minimum* i *Larsen proizvod*. *Mamdani minimum* [70] metoda, kao jedna od najčešće korišćenih, zasniva se na *Mamdani* pravilima, i pogodna je za rezonovanje nad leksičkim terminima, definisanim nad fuzzy skupovima. Osnova rezonovanja su fuzzy asocijativne matrice, koje se formiraju strukturu fuzzy pravila, definisanjem njegove premise i konsekvence.

Radi objašnjenja, navodimo jednostavan primer fuzzy pravila, koje manipuliše leksičkim terminima, i ima po jedan termin u premisi i konsekvenci (Izrazi 5 i 6).

$$X = \{(0, 0.0), (25, 0.5), (50, 1.0), (75, 0.5), (100, 0.0)\} \quad (5)$$

$$Y = \{(60, 0.0), (65, 0.5), (70, 1.0), (75, 0.5), (80, 0.0)\} \quad (6)$$

Nad varijablom *tacnost* definisan je termin *srednja*, koji je opisan trouglastim fuzzy skupom *X*. Nad varijablom *rezultat* definisan je termin *dobar*, koji je opisan trouglastim fuzzy skupom *Y*. Fuzzy skupovi su definisani uređenim parovima vrednosti i funkcija pripadanja. Korisnik sistema zatim definiše jedno pravilo (Izraz 7).

$$\text{IF } \textit{tacnost} = \textit{“srednja”} \text{ THEN } \textit{rezultat} = \textit{“dobar”} \quad (7)$$

Pošto su premisa i konsekvencija pravila fuzzy skupovi, sistem za rezonovanje formira asocijativnu matricu fuzzy pravila (Ilustracija 78). Svaki element matrice predstavlja preslikavanje funkcije pripadanja ulazne vrednosti (x_i) u funkciju pripadanja izlazne vrednosti (y_i). Kod *Mamdani minimum* metode, minimum predstavlja funkciju preslikavanja ulazne u izlaznu vrednost. Nakon konstrukcije matrice, fuzzy pravilo je spremno za korišćenje.

$$M_{Y|X} = \begin{array}{c|cccc} & x1 \rightarrow y1 & x1 \rightarrow y2 & \dots & x1 \rightarrow y5 \\ \hline x2 \rightarrow y1 & & & & \\ x2 \rightarrow y2 & & & & \\ \dots & & & & \\ \dots & & & & \\ x5 \rightarrow y1 & & & & x5 \rightarrow y5 \end{array} = \begin{array}{c|cccc} & \text{Min}(0,0,0,0) & \text{Min}(0,0,0,5) & \dots & \text{Min}(0,0,0,0) \\ \hline & \text{Min}(0,5,0,0) & \text{Min}(0,5,0,5) & \dots & \dots \\ & \dots & \dots & \dots & \dots \\ & \dots & \dots & \dots & \dots \\ & \text{Min}(0,0,0,0) & \dots & \dots & \text{Min}(0,0,0,0) \end{array} = \begin{array}{c|ccccc} & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ \hline & 0.0 & 0.5 & 0.5 & 0.5 & 0.0 \\ & 0.0 & 0.5 & 1.0 & 0.5 & 0.0 \\ & 0.0 & 0.5 & 0.5 & 0.5 & 0.0 \\ & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{array}$$

Ilustracija 77: Formiranje asocijativne matrice u Mamdani fuzzy pravilu

Najjednostavniji slučaj je da na ulaz pravila (Izraz 7) dovedemo vrednost 25 fazifikovanu kao *crisp* fuzzy skup (*crisp* skup je opisan u poglavlju „Tipovi fuzzy skupova”). Poređenjem (*matching*-om) sa fuzzy skupom X (Izraz 5), ovoj *crisp* vrednosti se dodeljuje 0.5 kao vrednost funkcije pripadanja, tako da se dobija vektorska reprezentacija X' ulazne vrednosti (Izraz 8).

$$X' = \{(0, 0.0), (25, 0.5), (50, 0.0), (75, 0.0), (100, 0.0)\} \quad (8)$$

Korišćenjem *Kronecker* proizvoda (koji omogućava množenje matrica proizvoljnih veličina i dimenzija) [72], množe se funkcije pripadnosti ovog vektora sa asocijativnom matricom $M_{Y|X}$ (Ilustracija 78). Proizvod je izlazni fuzzy skup Y' , kao maksimum minimuma parova verovatnoća x_i' fuzzy skupa X' , i elemenata m_{ij} matrice $M_{Y|X}$ (Izraz 9).

$$Y' = \text{Max}\{\text{Min}(x_i', m_{ij})\} \quad (9)$$

U praksi je najčešći slučaj da u premisi (i u konsekvenci) postoji više kriterijuma za zaključivanje (vidi izraz 3). Fuzzy pravila se tada usložnjavaju tako što se za svaki ulazni i izlazni kriterijum (termini, ili fuzzy skup) definiše posebna asocijativna matrica. Za slučaj da postoji i ulaznih kriterijuma A , i j izlaznih kriterijuma C , onda pravilo ima $i*j$ asocijativnih matrica ($M_{C_1|A_1}, \dots, M_{C_j|A_i}$). U slučaju da se uvećava broj kriterijuma u pravilima, geometrijski raste i kompleksnost pravila izražena u dimenziji fuzzy asocijativne matrice.

Konačan zaključak C se dobija primenom logičkog operatora (konjunkcije, ili disjunkcije) korišćenog u premisi, na konsekvence pravila (Izraz 10).

$$C = [A_x \circ M_{C_x|A_x}] \wedge [A_y \circ M_{C_y|A_y}] \dots \wedge [A_i \circ M_{C_i|A_i}] = C_x \wedge C_y \dots \wedge C_j \quad (10)$$

Pravila mogu da se optimizuju u slučaju da se ulazne vrednosti fazifikuju u *crisp* fuzzy skupove. Asocijativne matrice postaju suvišne, a rezonovanje se odvija korišćenjem odgovarajuće funkcije za preslikavanje (Izrazi 11 i 12).

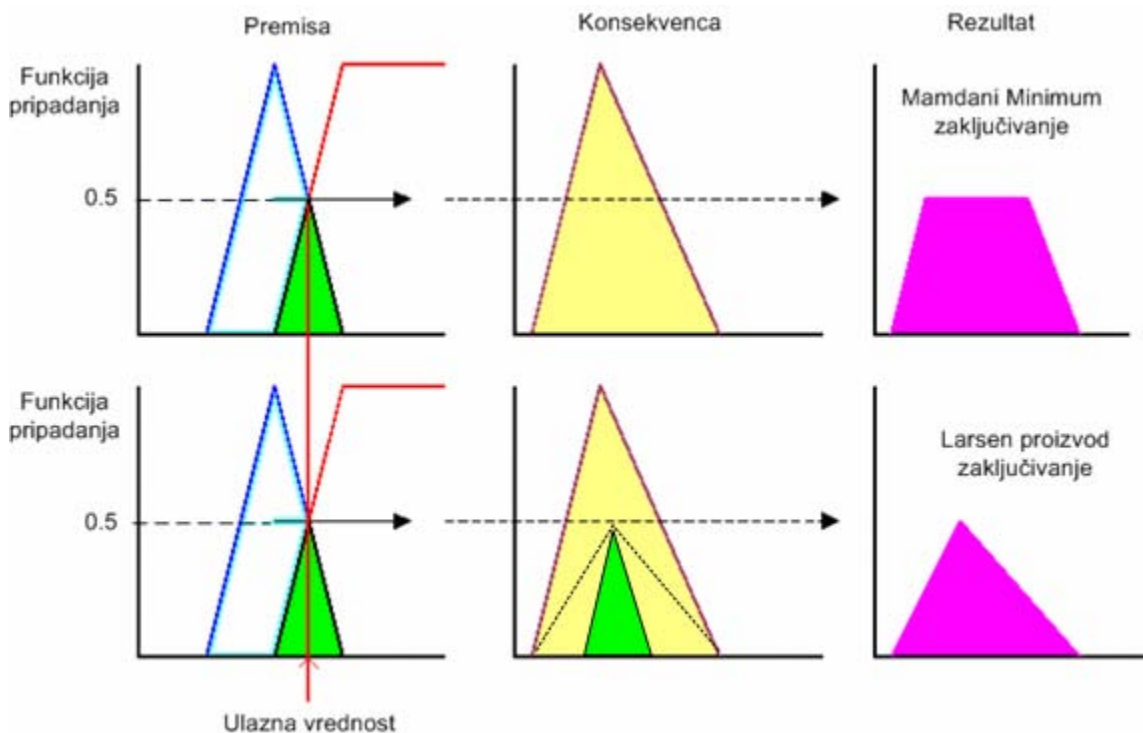
$$\min(\mu_a(x_i), \mu_b(y_j)) \wedge \mu_{c_n}(z) \quad (11)$$

$$\max(\mu_a(x_i), \mu_b(y_j)) \wedge \mu_{c_n}(z) \quad (12)$$

Ako su ulazni kriterijumi a i b , i ako se razmatraju diskretne ulazne vrednosti x_i za kriterijum a i y_j za kriterijum b , konjunkcija uslova u premisi, prenosi se na konjunkciju minimuma funkcija pripadanja ulaznih kriterijuma $\mu_a(x_i), \mu_b(y_j)$, sa funkcijama pripadanja fuzzy skupova zaključaka $\mu_{c_n}(z)$ (Izraz 11). Ako se radi o disjunkciji

kriterijuma premise, sistem vrši konjunkciju maksimuma ulaznih kriterijuma $\mu_a(x_i), \mu_b(y_j)$, sa funkcijama pripadanja fuzzy skupova zaključaka $\mu_{c_n}(z)$ (Izraz 12).

Larsen proizvod [71] metoda se zasniva na Mamdani pravilima i slična je Mamdani metodi. Ova metoda kao izlaz daje *skaliran* fuzzy skup, za razliku od *odsečenih* fuzzy skupova u konsekvencama kod *Mamdani minimum* rezonovanja (Ilustracija 79). Ako se zaključuje na osnovu ulaznih vrednosti, fazifikovanih kao *crisp* vrednosti, *Larsen proizvod* metoda funkcioniše po sledećem izrazu (Izraz 13).



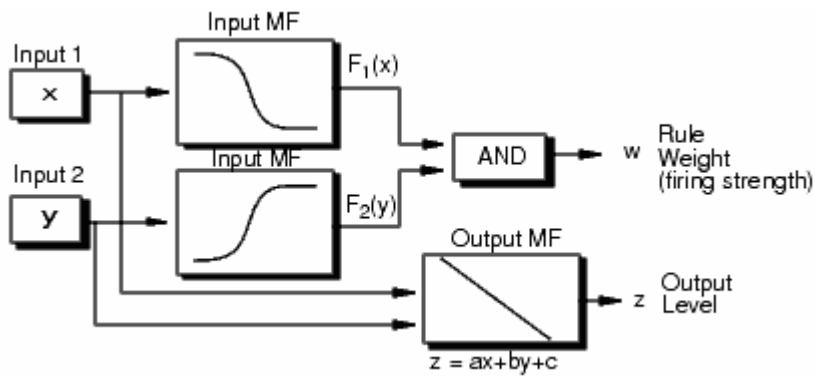
Ilustracija 78: Upoređivanje Mamdani minimum i Larsen proizvod zaključivanja

Kod *Larsen proizvod* metode, izlazni fuzzy skup C' se dobija kao proizvod (minimum) funkcija pripadanja premise $\mu_a(x)$ i konsekvence pravila $\mu_c(y)$ (Izraz 13).

$$C' = \mu_a(x) \cdot \mu_c(y) \quad (13)$$

U slučaju da postoji više kriterijuma u premisi, koristi se kombinovana funkcija pripadanja, dobijena na neki od prethodno opisanih načina (Izrazi 11 i 12). Interesan je i specifični slučaj kada se u konsekvencama pravila koristi *singleton* fuzzy skup. Tada se, kod obe prethodno opisane metode fuzzy rezonovanja, dobija identični rezultat.

Pored navedenih, postoji *Tsukamoto* metoda rezonovanja. U nekim izvorima ova metoda rezonovanja se naziva još i kao *Sugeno* metoda. Razlog ovoj činjenici je velika sličnost između *Tsukamoto* i *Takagi-Sugeno-Kang* pravila. U oba slučaja fuzzy rezonovanje (Ilustracija 80) se odvija kroz fazifikaciju ulaznih vrednosti (x, y) određivanjem funkcija pripadanja ($F_1(x)$ i $F_2(y)$), a zatim se računa težina pravila (w) na osnovu konjunkcije tih funkcija pripadanja.



Ilustracija 79: Sugeno metoda fuzzy rezonovanja

Izlazna vrednosti iz pravila (z) razmatra se zajedno sa težinom pravila. Konačni izlaz je diskretna vrednost, koja predstavlja težinsku sredinu svih izlaznih fuzzy vrednosti (Izraz 14).

$$r = \frac{\sum_{i=1}^N w_i z_i}{\sum_{i=1}^N w_i}, \quad z_i - \text{izlazna vrednosti pravila, } w_i - \text{težina (DOF) pravila} \quad (14)$$

Tsukamoto (Sugeno) metoda je pogodna za adaptivne sisteme, ali ne i za sisteme sa rezonovanjem na bazi leksičkih varijabli. Vrlo diskriminišuće je što u konsekvencama fuzzy pravila ne mogu da budu fuzzy skupovi, već isključivo diskretne vrednosti (konstante, ili rezultati izl.funkcije). Kod ove metode, proces defazifikacije je završen završetkom fuzzy rezonovanja, tako da pruža bolje mogućnosti optimizacije nego ostale navedene metode. Pored navedenih, postoje i druge metode fuzzy rezonovanja (npr.metoda drastičnog proizvoda i metoda ograničenog proizvoda).

5.4.3.6 Defazifikacija

Defazifikacija je poslednja faza fuzzy rezonovanja. Izuzev kod *Tsukamoto (Sugeno)* metode, zaključci fuzzy rezonovanja su rezultujući fuzzy skupovi, koji su najčešće (izuzev ako se predstavljaju *crisp* fuzzy skupovima) kompoziti osnovnih tipova. Da bi rezultati bili upotrebljivi, potrebno je da se konvertuju u odgovarajući diskretan oblik. Ovaj proces se naziva defazifikacija i obavlja se primenom odgovarajuće matematičke kalkulacije, kojom se fuzzy skup konvertuje u realan broj [73]. Postoje različite tehnike defazifikacije.

Centroid

Centroid, ili centar gravitacije je jedna od najčešće korišćenih metoda defazifikacije. Sličan princip je korišćen i kod *Tsukamoto (Sugeno)* metode rezonovanja (Izraz 14). Suština algoritma za defazifikaciju na osnovu centroida [74] je da se rezultujući fuzzy skup fragmentira (ili aproksimira) na linearne segmente, a zatim, da se za svaki segment računa posebno centar gravitacije. Rezultujuća vrednost predstavlja prvi momenat površine celog rezultujućeg skupa (Izraz 15).

$$x' = \frac{\sum_{i=1}^N x_i \cdot A_i}{\sum_{i=1}^N A_i} \quad (15)$$

Pojedinačni centri gravitacije su x_i , površine linearnih segmenata su A_i , a N je ukupan broj segmenata.

Sredina maskimuma

Metoda sredine maskimuma [74] zasniva se na izračunavanju aritmetičke sredine skupa izlaznih vrednosti za koje funkcija pripadanja u rezultujućem fuzzy skupu ima maksimalnu vrednost (Izraz 16).

$$x_{rez} = \frac{\sum_{i=1}^N x_i}{N}, \mu(x_i) = MAX \quad (16)$$

Ova, za implementaciju vrlo jednostavna metoda, ima mnoštvo nedostataka. Svi proističu iz činjenice da se ovom metodom gubi zavisnost izlazne vrednosti i funkcije pripadanja fuzzy skupu.

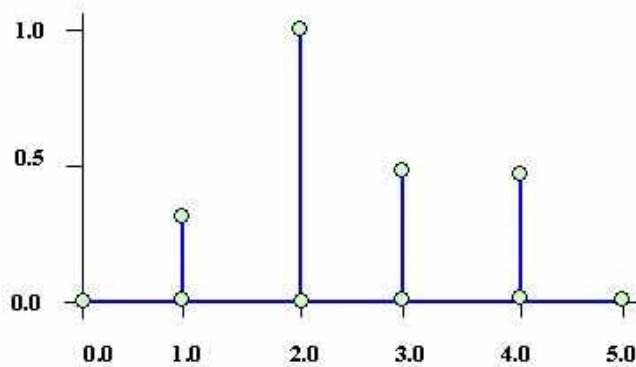


Ilustracija 80: Primer korišćenja metode sredine maksimuma

Ova pojava je najbolje ilustrovana prethodnom slikom (Ilustracija 81). Izlazni fuzzy skup je predstavljen plavom linijom. U skupu postoje dva maksimuma, za vrednosti x_1 i x_2 . Korišćenjem metode sredine maksimuma dobija se diskretna izlazna vrednosti x_{rez} , čija funkcija pripadanja ima vrednost 0 (ne postoji pripadanje izlaznom fuzzy skupu). Ako izlazni fuzzy skup nije konveksan i ako postoji velika disperzija rezultata, vrlo je verovatna pojava ovakvog slučaja.

Metod težinske sredine

Metod težinske sredine je pogodan kada je rezultujući fuzzy skup kompozit *crisp* fuzzy skupova (Ilustracija 82). To je slučaj kod *Takagi-Sugeno-Kang* pravila, odnosno *Tsukamoto* (*Sugeno*) metode rezonovanja. Jedan od problema metode sredine maksimuma je da iz razmatranja isključuje sve vrednosti čije su funkcije pripadanja manje od maksimalne vrednosti.



Ilustracija 81: Slučaj *crisp* fuzzy skupova kao rezultata rezonovanja

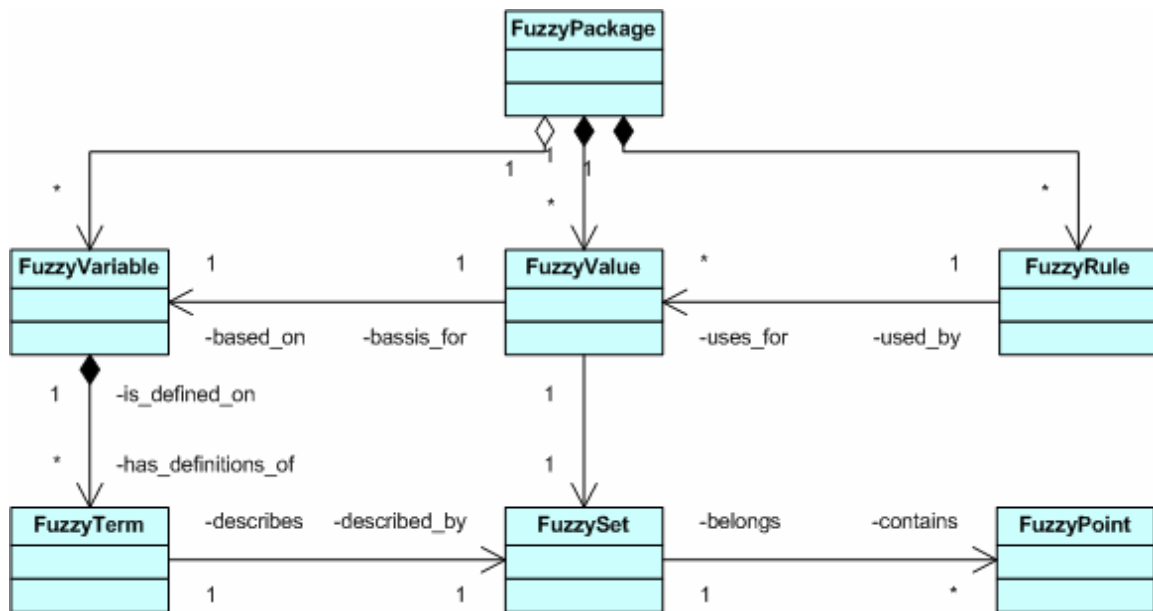
Kada je rezultujući fuzzy skup kompozit *crisp* fuzzy skupova, metod centroida (pošto se zasniva površinama fragmenata aproksimacije), takođe ne daje adekvatan rezultat. Metoda težinske sredine, zasnovana na jednačini (Izraz 14) daće najadekvatniji rezultat – pravila koja imaju najveću DOF vrednost (poglavlje 5.4.3.4), ispoljiće najveći uticaj na defazifikovanu vrednost (praktično, vrednost koja se očitava na apscisi izlaznog dijagrama).

5.4.5 Paketi fuzzy baze znanja

Da bi se omogućila adaptacija sistema prema navedenim karakteristikama, i da bi se omogućila implementacija različitih pedagoških aspekata ocenjivanja, korišćenjem fuzzy logike, dizajnirani su posebni moduli koji omogućavaju višekriterijumsko rezonovanje, i implementaciju različitih sistema ocenjivanja. *FuzzyOntology*, *FuzzyReasoner* i *FuzzyEvaluator* su kohezivni moduli kojima su implementirane navedene funkcionalnosti.

5.4.5.1 Paket FuzzyOntology

Koncepti predstavljeni u prethodno poglavlju omogućavaju neegzaktno rezonovanje u procesu ocenjivanja. Oni su predstavljeni kao klase paketa *FuzzyOntology* (Ilustracija 83), koje čine fuzzy bazu znanja. Osnovne klase paketa su fuzzy varijabla (*FuzzyVariable*) i fuzzy vrednost (*FuzzyValue*). Fuzzy varijable (poglavlje 5.4.3.1) enkapsuliraju svojstva iz realnog sveta o kojima se rezonuje na osnovu fuzzy logike. Fuzzy varijablama su predstavljeni kriterijumi za ocenjivanje studenata (npr. *tačnost*, *brzina*, *netačnost*, *rezultat*). Fuzzy vrednosti se definišu nad fuzzy varijablama, koristeći odgovarajući fuzzy skup. Fuzzy varijable su sistemski resurs, što znači da su deljive među različitim fuzzy paketima (klasa *FuzzyPackage*). U jednom fuzzy paketu nalaze se definicije konceptata (fuzzy varijabli, vrednosti i pravila) potrebne za jednu fuzzy bazu znanja. Sistem omogućava da se za svaku proveru definiše posebna fuzzy baza znanja.

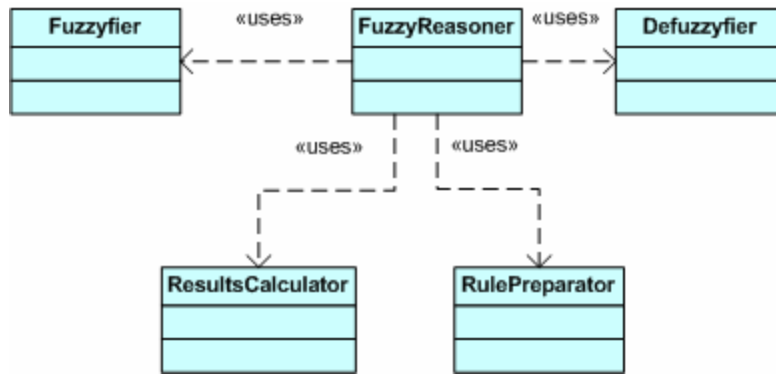


Ilustracija 82: Ontologija fuzzy baze znanja

Fuzzy pravila (klasa *FuzzyRule*) se sastoje od fuzzy vrednosti različitih varijabli, dovedenih u vezu različitim operatorima (logičkim ili skupovnim operatorima). Isto kao kod pravila u diskretnoj (bivalentnoj) logici, ona imaju premisu i akcioni deo. U fuzzy pravilima se vrši preslikavanje fuzzy skupova vrednosti u premisi u fuzzy skupove vrednosti koje se nalaze u zaključku pravila (poglavlje 5.4.3). Fuzzy termini (klasa *FuzzyTerm*) služe da leksički opišu vrednosti fuzzy varijabli. Isto kao fuzzy vrednosti, definišu se fuzzy skupom. Terminima se omogućuje leksičko rezonovanje u sistemu (prijemčivije ljudima). Svaki fuzzy skup je opisan nizom tačaka (*FuzzyPoint*) – uređenim parovima vrednosti varijable i funkcije pripadanja. Za razliku od fuzzy varijabli, fuzzy vrednosti i pravila nisu deljiv resurs. Ovi entiteti pripadaju samo jednoj fuzzy bazi znanja – agregiraju u fuzzy paket.

5.4.5.2 Paket *FuzzyReasoner*

FuzzyReasoner predstavlja glavnu klasu ovog paketa. Ova klasa koristi sve koncepte (klase) fuzzy ontologije (poglavlje 5.4.5.1). Klasa *FuzzyReasoner* omogućava napredno ocenjivanje korišćenjem različitih kriterijuma (poglavlje 5.4.4). Da bi se izbegla kompleksnost koju nameću faze fuzzy rezonovanja (poglavlje 5.4.3), ovaj paket sadrži nekoliko pomoćnih klasa za implementaciju pojedinih faza, okupljenih oko klase *FuzzyReasoner* (Ilustracija 84).



Ilustracija 83: Klase paketa FuzzyReasoner

Pored klase *ResultsCalculator* postoje četiri pomoćne klase:

- *ResultsCalculator* – klasa namenjena pripremi ulaznih podataka: statistička obrada i normalizacija pogodna za fuzzy rezonovanje
- *Fuzzyfier* – klasa namenjena fazifikaciji ulaznih podataka. Izdvajanjem posebne klase dobila se mogućnost eksplicitnog definisanja tipa fuzzy skupa u koji se transformišu ulazne vrednosti.
- *RulePreparator* – klasa namenjena pripremi fuzzy pravila. Pored kriterijuma u premisi i konsekvenci, fuzzy pravila treba da se podese za način kombinovanja ulaznih kriterijuma i način egzekucije pravila (poglavljje 5.4.3.5). Na kraju, pravila se pune fazifikovanim ulaznim vrednostima.
- *Defuzzyfier* - klasa namenjena defazifikaciji izlaznih fuzzy vrednosti – proizvoda fuzzy rezonovanja. Izdvajanjem posebne klase dobila se mogućnost eksplicitnog definisanja tehnike koja će se koristiti u defazifikaciji (poglavljje 5.4.3.6).

Sve metode ovih pomoćnih klasa su statičke, tako da su relacije su povezane sa klasom *ResultsCalculator* stereotipnom zavisnošću *uses*. Svi podaci kojima pomoćne klase operišu su tranzijentni, postoje samo u njihovim statičkim metodama u vidu lokalnih varijabli. Korišćenjem funkcionalnih poziva zajedničkih (statičkih) metoda klasa, značajno je optimizovano korišćenje memorije i platforme za procesiranje (Java virtualne mašine). Druga prednost je fleksibilnost i proširljivost mašine za rezonovanje – mogućnost dodavanja novih algoritama i tehnika u svakoj fazi rezonovanja posebno.

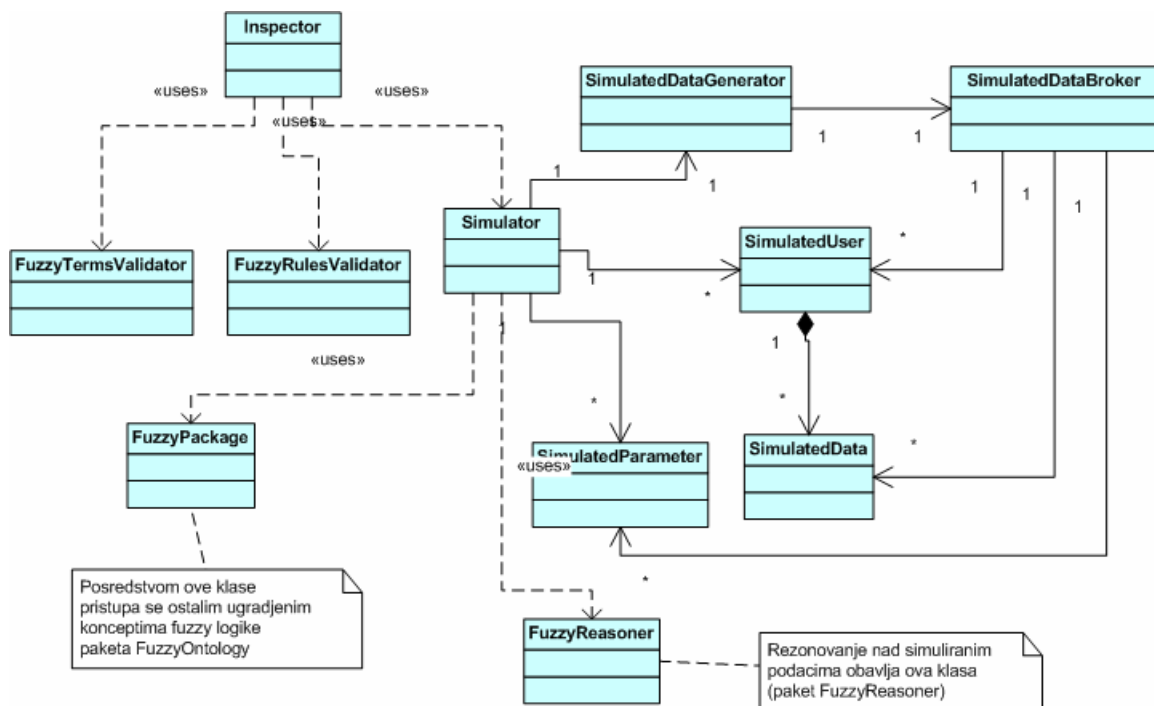
5.4.5.3 Paket FuzzyEvaluator

Klase ovog paketa imaju dvojaku namenu:

- Provera korektnosti definicija fuzzy baze znanja – fuzzy varijable, skupovi, termini i pravila (poglavljje 5.4.3.1) se komponuju dinamički, na osnovu unosa nastavnika. Da bi sistem mogao pouzdano da funkcioniše, potrebno je da u fuzzy bazi znanja ne bude nedefinisanih elemenata. Na primer, može da se desi slučaj da nastavnik ne obuhvati definicijama (fuzzy) termina celokupan skup vrednosti nad kojima je definisan kriterijum (fuzzy varijabla). Drugi propust može da se desi ako nisu sve fazifikovane ulazne vrednosti pokrivene premisama (fuzzy)

pravila. U tom slučaju, proces rezonovanja dolazi u neregularno stanje, a sistem nema adekvatan odziv mašine za zaključivanje.

- Provera rezonovanja korišćenjem simulacionih podataka – pre upotrebe u stvarnim uslovima, najsigurnija provera mašine za zaključivanje je da se obavi njeno korišćenje nad simuliranim podacima. Na osnovu definisanih koncepata fuzzy baze znanja, sistem generiše diskretne podatke u opsezima realnih vrednosti (koje iščitava iz studentskih rezultata i performansi testiranja). Oni predstavljaju ulaze u fuzzy mašinu za zaključivanje. Obavlja se rezonovanje nad simulacionim podacima, snimaju se rezultati i performanse. Ako su svi izlazni podaci korektni i dobijeni u granicama vremenske tolerancije, sistem daje indicaciju da može da se obavi napredno ocenjivanje.



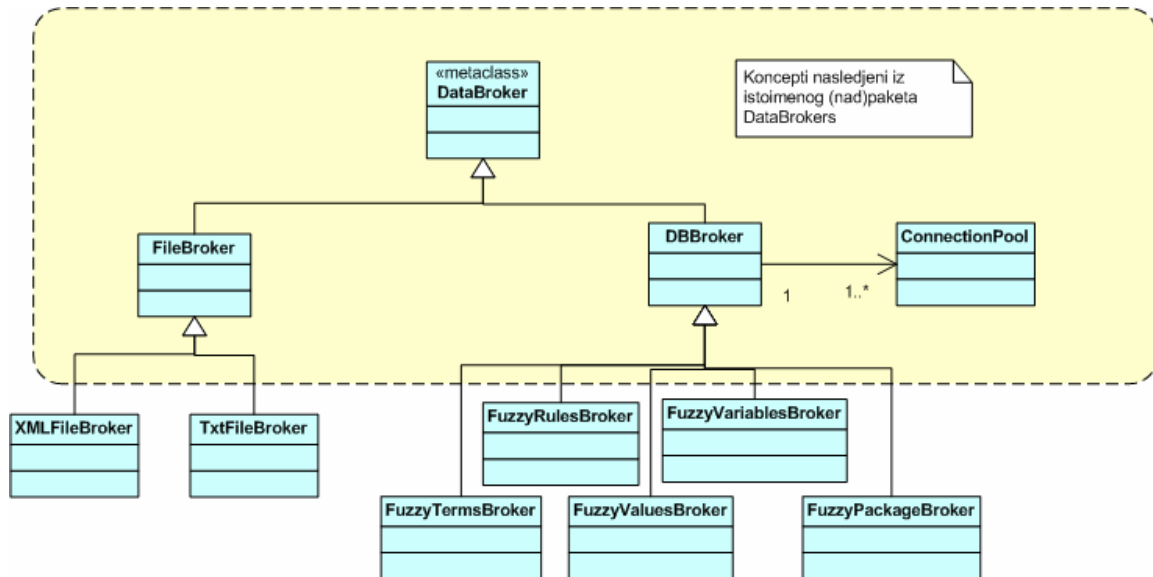
Ilustracija 84: Koncepti (klase) paketa *FuzzyEvaluator*

Klasa *Inspector* implementira projektni obrazac *Facade* [113]. Ona koristi klase namenjene za evaluaciju fuzzy termina i fuzzy pravila (*FuzzyTermsValidator* i *FuzzyRulesValidator*). Isto tako koristi klasu *Simulator* namenjenu za simulaciju rezonovanja. *Simulator* koristi klasu *FuzzyPackage* iz paketa *FuzzyOntology* (poglavlje 5.4.5.1) i *FuzzyReasoner* iz istoimenog paketa (poglavlje 5.4.5.2). Klasa *SimulatedDataGenerator* namenjena je generisanju simulacionih podataka. Klasa *SimulatedDataBroker* namenjena je pamćenju i učitavanju simulacionih podataka. Klase *SimulatedUser*, *SimulatedParameter* i *SimulatedData* su pomoćne klase namenjene za čuvanje podataka simulacije u radnoj memoriji.

5.4.5.4 Paket *DataBrokers*

Prethodno opisani paketi *FuzzyOntology* i *FuzzyReasoner* sadrže klase koje se serijalizuju. Tako da svaka od klasa ima svog posrednika prema BP – istoimenu

brokersku klasu. Sve brokerske klase fuzzy baze znanja (izuzev *SimulatedDataBroker*) se nalaze grupisane u paketu *DataBrokers*.



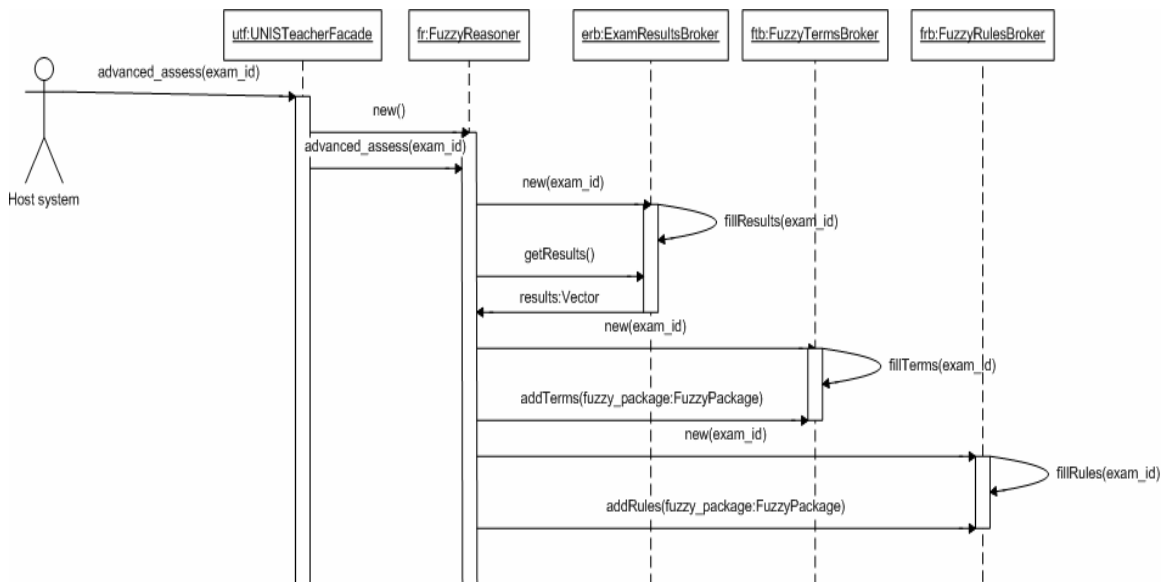
Ilustracija 85: Koncepti (klase) paketa *DataBrokers*

Da bi se izbeglo redundantno definisanje istih koncepata, kao roditeljske klase za brokere fuzzy koncepata korišćene su klase istoimenog paketa na jednom nivou više (poglavlje 5.4.2.4). Rezultujući model sadrži hijerarhijsku strukturu na čijem vrhu je apstraktna klasa *DataBroker* koja ima jedinu namenu da poveže dve raznorodne klasne strukture – za rad sa BP (*DBBroker*) i za rad sa fajl sistemom (*FileBroker*). Za rad sa resursima učenja projektovana je klasa *FileBroker*. Ova klasa da implementira primitive za rad sa fajlovima (kreiranje, preimenovanje, brisanje, premeštanje i zatvaranje fajlova). Sve ostale metode se definišu u izvedenim klasama (*XMLFileBroker* i *TxtFileBroker*).

5.4.6 Dijagrami interakcije

5.4.6.1 Fuzzy rezonovanje

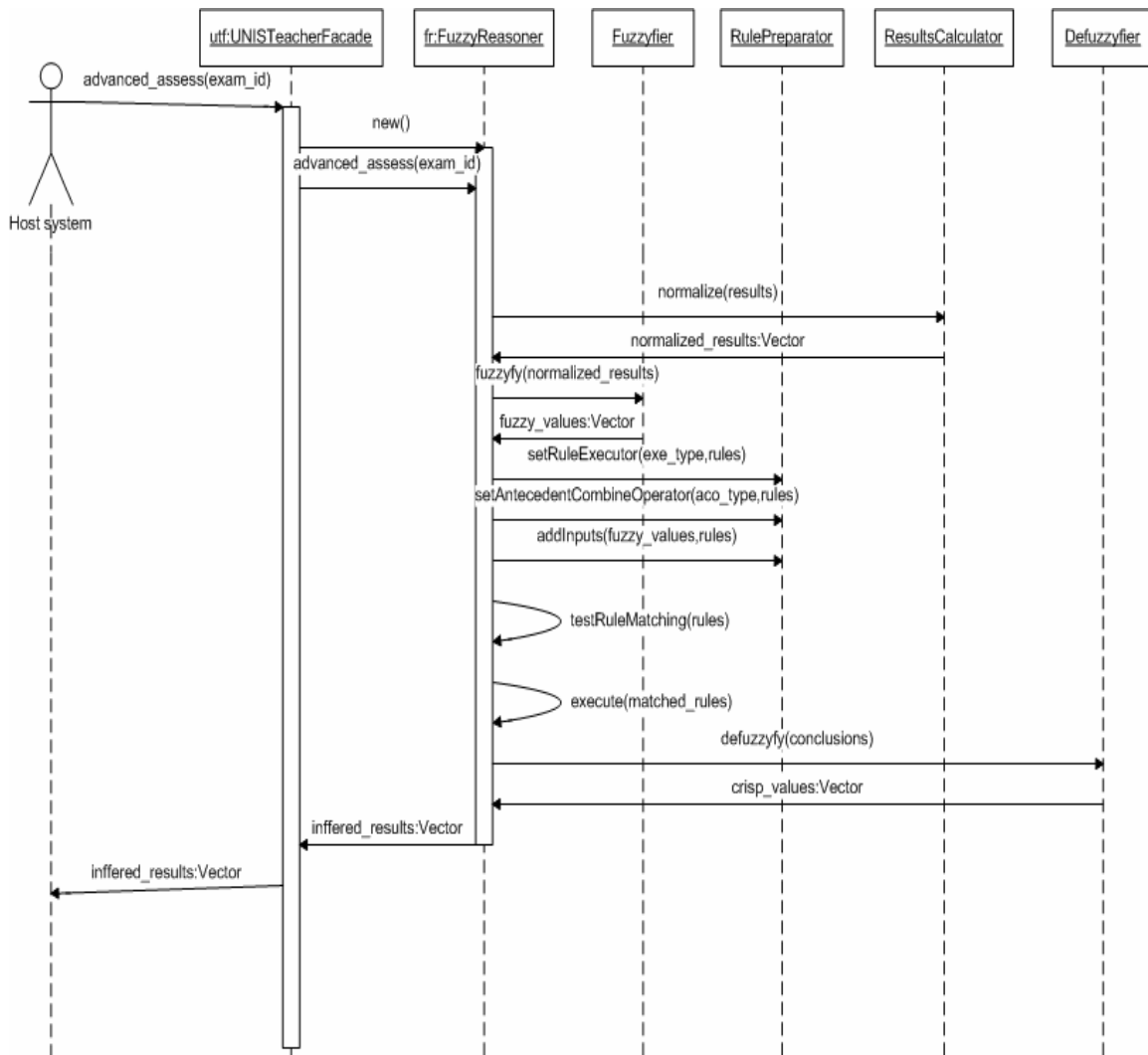
Proces fuzzy rezonovanja se odvija u dva ključna koraka. Prvi je korak (Ilustracija 87) obuhvata prikupljanje svih potrebnih podataka za zaključivanje. Sistem domaćina inicira ovu aktivnost se pozivom metode *advanced_assess* objekta *utf* klase *UNISTeacherFacade* (poglavlje 5.4.2). Kao argument metode se šalje identifikator provere koja treba da se napredno oceni. U propagaciji metode kroz sistem najpre se instancira objekat *fr* klase *FuzzyReasoner* istoimenog paketa (poglavlje 5.4.5.2). Ovo je glavna klasa za rezonovanje (ista se koristi i u evaluaciji fuzzy baze znanja – poglavlje 5.4.6.2).



Ilustracija 86: Prvi korak fuzzy rezonovanja

Rezonovanje započinje pozivom istoimene metode (*advanced_assess*) objekta *fr*. Pomoću *ExamResultsBroker* objekta, iz BP se dobijaju rezultati studenata (vektor *results*) iz željene provere. Nakon toga se pomoću *FuzzyTermsBroker* objekta, iz BP se uzimaju potrebni fuzzy skupovi (termini) kojima će se manipulirati u procesu rezonovanja. Klasa *FuzzyReasoner* u sebi sadrži objekat *fuzzy_package* klase *FuzzyPackage* (poglavlje 5.4.5.1), a koji u sebi može da enkapsulira kompletnu fuzzy bazu znanja. Pozivom metode *addTerms* klase *FuzzyTermsBroker*, kojoj se prosleđuje objekat *fuzzy_package*, *fr* objekat se puni terminima. Pozivom metode *addRules* klase *FuzzyRulesBroker*, kojoj se prosleđuje objekat *fuzzy_package*, *fr* objekat se puni pravilima.

Nakon što su prikupljeni ulazni podaci i podaci fuzzy baze znanja, prelazi se na drugi korak rezonovanja. Objekat *fr* klase *FuzzyReasoner* prosleđuje studentske rezultate klasi *ResultsCalculator* pozivom njene metode *normalize*. Podaci se statistički obrađuju i normalizuju, kako bi njihovo variranje bilo u granicama definisanim fuzzy bazom znanja. Objekat *fr* dobija normalizovane podatke kao povratnu vrednost metode *normalize* (vektor *normalized_data*).



Ilustracija 87: Drugi korak fuzzy rezonovanja

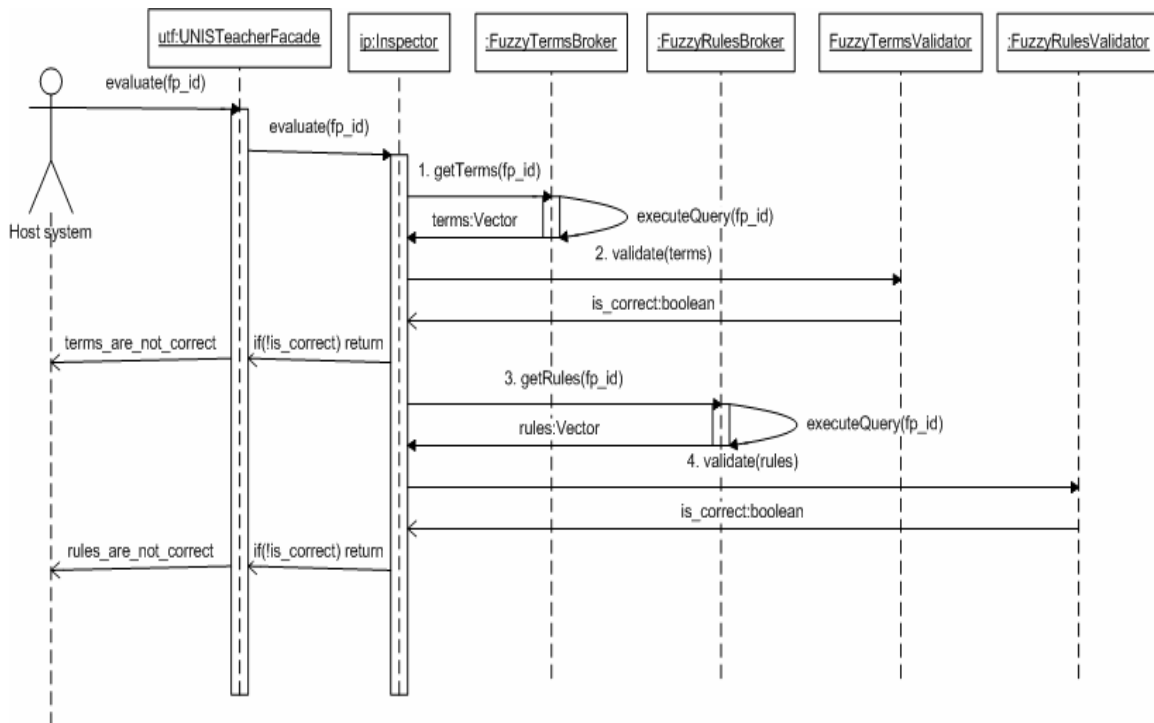
Sledi dizajn prethodno objašnjenih faza fuzzy rezonovanja (poglavlja 5.4.3.3 – 5.4.3.6). Objekat *fr* zatim prosleđuje normalizovane podatke klasi *Fuzzyfier* pozivom njene statičke metode *fuzzyfy*. Kao rezultat ove metode dobijaju se fazifikovani podaci (vektor *fuzzy_values*). Zatim se pomoću klase *RulePreparator* pripremaju fuzzy pravila za rezonovanje. Podešava se egzekutor (metod *setRuleExecutor*), operator za kombinovanje ulaznih podataka (metod *setAntecedentCombineOperator*) i na kraju se pravilima dodaju ulazni podaci – *fuzzy_values*.

Samo zaključivanje realizuje se u dva koraka i obavlja se u objektu *fr* klase *FuzzyReasoner*. Najpre se biraju pravila koja će da se izvrše (metoda *testRuleMatching*), a zatim se odabrana pravila (vektor *matched_rules*) izvršavaju (metoda *execute*). Rezultat svakog pravila je fuzzy vrednost. Ove vrednosti zajedno čine rezultujući fuzzy skup, koji predstavlja zaključak o rezultatu jednog studenta. Svi fuzzy skupovi se smeštaju u vektor – *conclusions*, koji se prosleđuje statičkoj metodi *defuzzyfy* klase *Defuzzyfier*, da bi se dobile diskretne vrednosti rezultata (vektor *crisp_values*). Vektor sa rezultatima studenata se vraća preko instance *UNISTeacherFacade* sistemu domaćinu (klijentu).

5.4.6.2 Evaluacija fuzzy baze znanja

Evaluacija fuzzy baze znanja se realizuje u dva koraka. U prvom se proveravaju definicije fuzzy termina i pravila. Ako se ova provera uspešno završi, u drugom koraku se proverava rad sistema – rezonovanje nad simuliranim podacima.

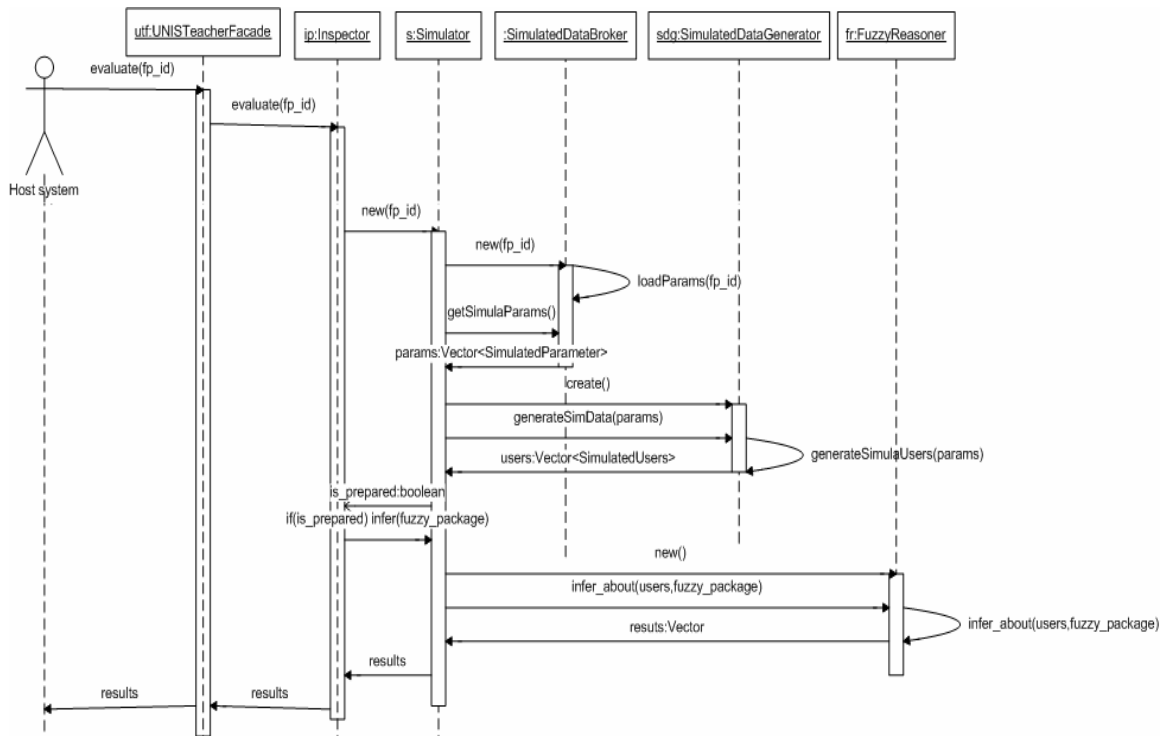
Zahtev za evaluaciju fuzzy baze znanja se dobija od sistema domaćina (Ilustracija 89). Ovaj zahtev u vidu funkcionalnog poziva prihvata objekat klase *UNISTeacherFacade* (poglavlje 5.4.2) i prosleđuje ga *Facade* [113] klasi *Inspector*. Na osnovu evaluacionog zahteva, čiji argument je identifikator fuzzy paketa koji se evaluira (poglavlje 5.4.5.1), *Inspector* se puni najpre terminima (poziv metode 1 – *getTerms*), a ako uspe provera termina – i pravilima (poziv metode 3 – *getRules*). Ako termini nisu validni (poziv statičke metode 2 – *validate* u klasi *FuzzyTermsValidator*, kojoj se prosleđuje vektor termina), provera se završava vraćanjem poruke sistemu domaćina – *terms_are_not_correct*. Inače, provera se nastavlja pozivanjem istoimene statičke metode 4 – *validate* u klasi *FuzzyRulesValidator*. U slučaju da pravila nisu korektno definisana, validacija se prekida vraćanjem poruke *rules_are_not_correct* sistemu domaćina.



Ilustracija 88: Prvi korak evaluacije fuzzy paketa

Ako provere ove dve klase završe uspešno, u sledećem koraku se prelazi na simulaciju rezonovanja (Ilustracija 90). Klasa *Inspector* najpre instancira objekat klase *Simulator*. Ovaj objekat konstruiše brokerski objekat prema BP (klasa *SimulatedDataBroker*), prosleđujući mu identifikator fuzzy pakovanja. Broker koristi prosleđeni parametar da u fazi konstrukcije napuni vektor simulacionih parametara, koji se vraća simulatoru na korišćenje (kao povratna vrednost metode *getSimulaParams*). Zatim objekat klase *Simulator* instancira objekat klase *SimulatedDataGenerator* (označen kao *sdg*), nakon čega poziva njegov metod *generateSimData* prosleđujući vektor simulacionih parametara. Ovaj poziv inicira izvršenje interne metode *generateSimulaUsers* kojoj se

prosleđuje vektor parametara. Konačno, metod *generateSimData* vraća simulatoru vektor simuliranih podataka korisnika (instance *SimulatedUser*), napunjenih simuliranim podacima (instance *SimulatedData*).



Ilustracija 89: Drugi korak evaluacije fuzzy paketa

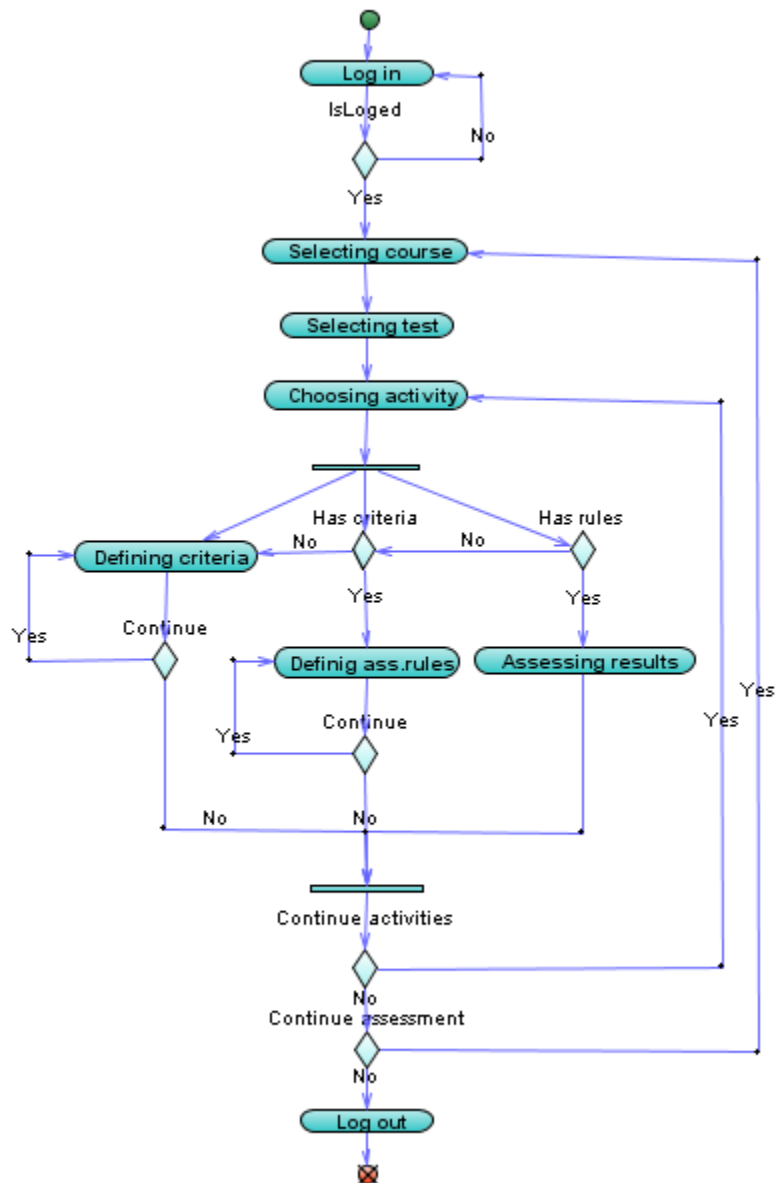
Ako je sve uredi, *Simulator* objekat šalje poruku *Inspector*-u da je sve spremno za simulaciju. *Inspector* objekat u svakom slučaju proverava vraćenu vrednost i ako je sve regularno poziva metod *infer* klase *Simulator*, koji najpre instancira objekat *fr* klase *FuzzyReasoner* istoimenog paketa (poglavlje 5.4.5.2). Zatim *Simulator* na objektu *fr* poziva metod *infer_about*, prosleđujući fuzzy bazu znanja (enkapsuliranu u klasi *FuzzyPackage* – poglavlje 5.4.5.1) i simulirane podatke (vektor *users* objekata *SimulatedUser*). *Simulator* prihvata rezultate rezonovanja u vidu vektora *results*, koje prosleđuje *Inspector*-u, a ovaj objekat vraća isti sistemu – domaćinu na interpretaciju i analizu.

5.4.7 Korišćenje sistema: modelovanje Fuzzy baze znanja

Jedan od ključnih problem koji je trebalo da se reši je da nastavnik bude u stanju da manipuliše fuzzy konceptima bez potrebe ikakvog znanja o tom domenu. Trebalo je da se napravi alat, koji bi nastavniku omogućio da svoje iskustvo iz ocenjivanja lako prenese na sistem, i da mu sistem za uzvrat omogući višekriterijumsko ocenjivanje rezultata testova.

5.4.7.1 Dijagram aktivnosti

Na sledećem dijagramu aktivnosti (Ilustracija 91) predstavljen je proces modelovanja fuzzy baze znanja za potrebe naprednog ocenjivanja.



Ilustracija 90: Dijagram aktivnosti za naprednog ocenjivanja

Nastavnik, nakon logovanja, bira predmet i proveru na koju želi da primeni napredno ocenjivanje. Definisanje naprednog ocenjivanja obavezno započinje izborom kriterijuma ocenjivanja. Podržani su kriterijumi koji se beleže u sistemu. Na proveru, inicijalno se beleži vreme, tačnost i netačnost studentskih odgovora. Ne moraju svi navedeni kriterijumi da budu uključeni u ocenjivanje. Zajedno sa kriterijumima, definišu se i termini kojima se manipuliše u ocenjivanju. Broj termina po kriterijumu ocenjivanja je proizvoljan.

Nakon definisanja kriterijuma i termina, nastavnik treba da definiše pravila za ocenjivanje. Sistem neće dozvoliti definisanje pravila ako ne postoji barem dva kriterijuma. Kriterijumi za ocenjivanje predstavljaju nezavisne varijable i sistem ih

isključivo koristi za premise pravila. Ocene predstavljaju termine izlaznog kriterijuma (zavisna varijabla), tako da formiraju konsekvence pravila.

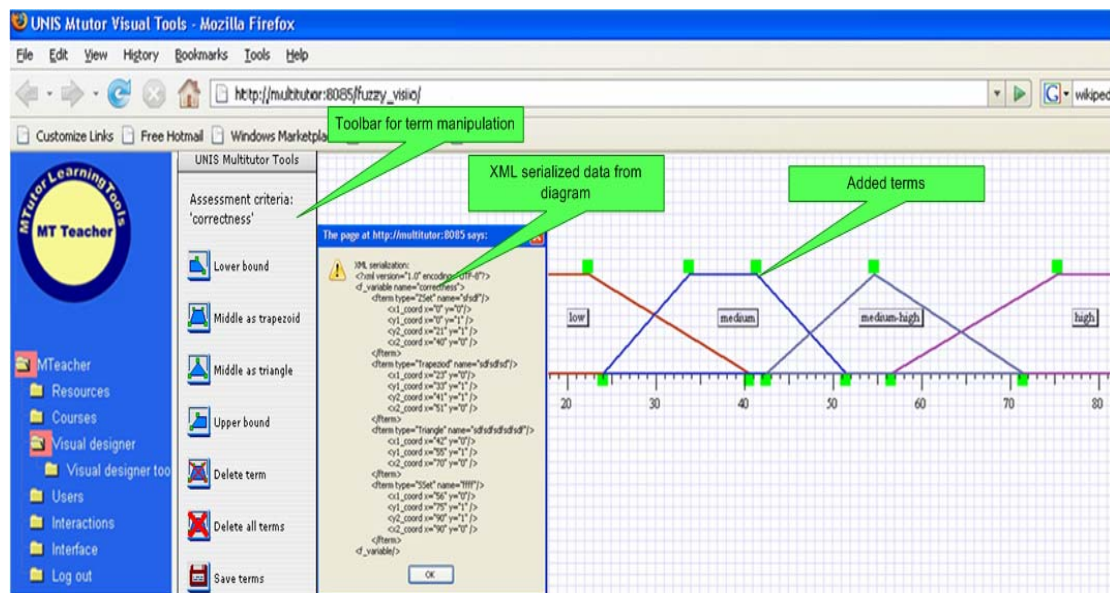
$$N_r = \prod_{i=1}^{N_c} N_t \quad (17)$$

Broj pravila za ocenjivanje N_r (izraz 17), koje treba definisati, jednak je proizvodu broja kriterijuma N_c i broja termina koji su definisani unutar svakog kriterijuma ponaosob N_t . To znači da nastavnik treba da bude svestan da kompleksnost rezonovanja (broj pravila) zavisi direktno od broja definisanih termina po uključenim kriterijumima.

Ocenjivanje rezultata proveru predstavlja zavisnu aktivnost, koja može da se realizuje tek pošto su prethodne dve opisane aktivnosti završene. Nastavnik može da izvrši proveru definisanih koncepata (poglavlje 5.4.6.2) pre njihovog korišćenja na pravim rezultatima proveru. Izvorni podaci za zaključivanje ostaju nepromenjeni u toku i posle završenog rezonovanja (ocenjivanja).

5.4.7.2 Korisnički interfejs za definisanje termina

Pri dizajnu nastavničkog interfejsa, za definisanje fuzzy baze znanja za potrebe naprednog ocenjivanja, vodilo se računa da isti bude intuitivan i da ne zahteva od nastavnika razumevanje koncepata fuzzy logike, niti posebnu stručnost u radu računarom. Iz navedenih razloga kreiran je grafički alat koji bi omogućio nastavniku da vizuelizuje koncepte i termine koje će koristiti u naprednom ocenjivanju (Ilustracija 92).



Ilustracija 91: Izgled vizuelnog alata modelovanje fuzzy baze znanja

Kao što je opisano dijagramima aktivnosti (poglavlje 5.4.7.1), nakon izbora kriterijuma, nastavnik definiše termine za ocenjivanje. S leve strane radne površine se nalazi paleta sa funkcionalnom dugmadi (*Toolbar for term manipulation*). Termini se dodaju kao, minimalni (levi granični), maksimalni (desni granični), i kao srednji (trouglovi ili trapezoidi). Izborom (levim klikom mišem) termina određenog tipa, pojavljuje se

modalni dijalog koji zahteva od nastavnika da unese naziv termina. Sistem proverava unos, tako da ne može da se definiše termin istog imena za isti kriterijum. Nakon potvrde unosa, termin se pojavljuje kao geometrijski oblik na dijagram kriterijuma (radni panel sa koordinatnom mrežom), označen unešenim imenom (na primeru su predstavljena četiri termina: *low*, *medium*, *medium-high* i *high* kriterijuma *correctness*). Ovi trapezoidi i trouglovi predstavljaju fuzzy skupove. Dijagram kriterijuma na apscisi sadrži definisani opseg vrednosti kriterijuma (*universe of discourse*), a na ordinati funkciju pripadanja (*grade of membership*) (poglavlje 5.4.3.1). Uglovi fuzzy skupova termina (zeleni kvadrati) su pokretni i mogu se pomoću miša pomerati po x-osi. Na taj način se menja oblik fuzzy skupa termina, njegove pojedinačne vrednosti i pridružene funkcije pripadanja.

Kao što se iz primera može videti, alat dozvoljava da se vrednosti termina preklapaju, što je nezamislivo kod sistema zasnovanih na diskretnoj logici. Nastavnik dakle ne mora da zna jasne granice između, na primer, studenata koji su proveru uradili *brzo*, *srednje*, ili *sporo*. Na taj način mu je prepuštena potpuna sloboda u prenošenju vlastitog iskustva u ocenjivanju na sistem. Nastavnik može da uklanja termine, dodaje nove, i da pamti svoje unose, korišćenjem dugmadi na *toolbar*-u. Pored podataka za rezonovanje, a za potrebe vizualizacije, pamte se tačne koordinate i tipovi fuzzy skupova termina. Pri pritisku na dugme za pamćenje, ovi podaci se prikazuju u XML formatu, predstavljenom na posebnom panelu (*XML serialized data from diagram*).

5.4.7.3 Implementacioni detalji vizuelizacije fuzzy baze znanja

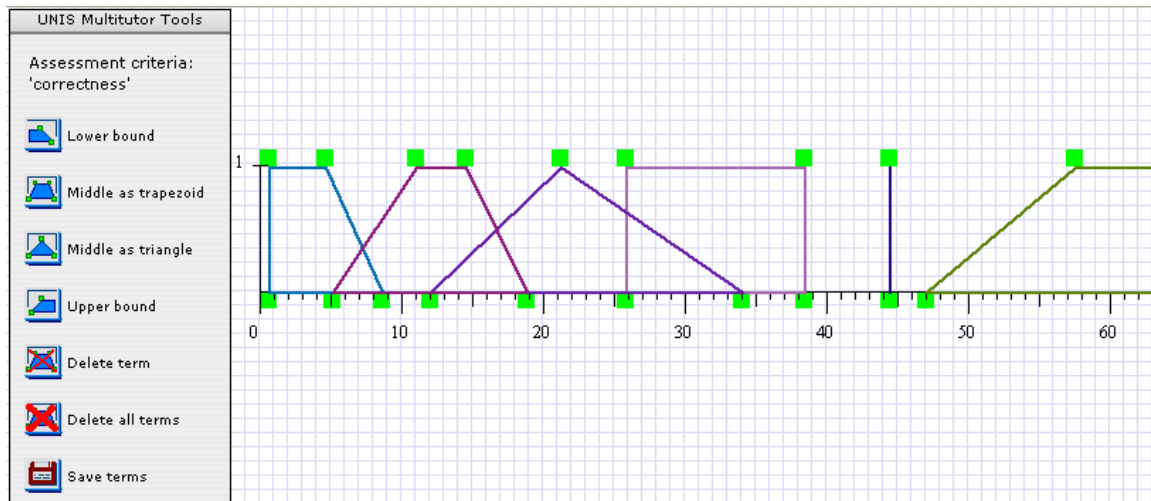
Vizualizator je aplikacija koja se izvršava na strani klijenta, u prozoru Web pretraživača. Implementirana je u skript jeziku (JavaScript), korišćenjem objektnog orijentisanog pristupa. U implementaciji su definisane klase za crtanje i klase interfejs kontrola, izvođenjem iz *Draw2D* [114] biblioteke.

```
var workflow = new draw2d.Workflow("paintarea");
var xstart=250;var ystart=100;var xend=1150;var yend=200;
var palette=new draw2d.FuzzyPalette("<%=fvb.getFuzzyVarName(2)%>",
    xstart,ystart,xend,yend,"<%=fterms.size()%>");
workflow.setToolWindow(palette);
palette.setPosition(10,10);
```

Fragment 6: Dodavanje palete sa alatima

Radna površina za crtanje predstavljena je varijablom *workflow*, instancom klase *Workflow* (Fragment 6). Svi objekti koji se dodaju na ovu površinu imaju relativne koordinate u odnosu na gornji levi ugao radne površine. Primer na prethodnoj ilustraciji prikazuje dodavanje palete, koja se puni dinamičkim sadržajem, na serverskoj strani, pre isporuke klijentu (crveni pravougaonici). Objekat *workflow* pozivanjem metode *setToolWindow* podešava prikaz palete na radnoj površini (naslovna linija, boja pozadine, fiksirane pozicije alata na paleti itd.).

Korišćenjem *Workflow* klase kao kontejnera za paletu i crtanje, omogućena je realizacija relativno jednostavnog rukovanja događajima korisničkog interfejsa (pritisak tastera miša, prevlačenje mišem i sl.). Svaki termin je predstavljen jednim trapezoidnim fuzzy skupom.



Ilustracija 92: Vrste fuzzy skupova koje podržava sistem

Sve ostale vrste fuzzy skupova (poglavlje 5.4.3.2) se izvode iz trapezoidnog (Ilustracija 93). Ako se poklapaju gornja temena trapezoida dobija se trouglasti fuzzy skup. Ako se poklapaju gornja i donja temena trapezoida dobija se *singleton* fuzzy skup. Ako trapezoid ima vertikalne bočne stranice, dobija se *Pi* fuzzy skup, ako je leva bočna strana vertikalna, dobija se desni linearni fuzzy skup, a ako je desna bočna strana vertikalna, dobija se levi linearni fuzzy skup.

Svako od temena fuzzy skupa predstavlja grafički objekat koji ima boju, dimenziju, tip objekta (kriterijum) koji predstavlja, 2D poziciju, ulazne i izlazne priključke (*port-ove*). Priključci prihvataju konektore (grafički – linije) za povezivanje temena. Aplikacija obrađuje događaje pomeranja (*drag*) i otpuštanja (*drop*) temena (Fragment 7).

```
draw2d.Figure.prototype.onDrag = function()
{
    if(this.x < this.left_bound_x || this.x > this.right_bound_x) return;
    if(this.y < this.down_bound_y || this.y > this.up_bound_y) return;
    this.x = this.draggable.getLeft();
    this.y = this.draggable.getTop();
    if(this.isMoving == false) {
        this.isMoving = true;
        this.setAlpha(0.5);
    }
    this.fireMoveEvent();
}
draw2d.Figure.prototype.onDragend = function()
{
    if(this.x < this.left_bound_x) this.setPosition(this.left_bound_x, this.y);
    if(this.x > this.right_bound_x) this.setPosition(this.right_bound_x, this.y);
    if(this.point_type == "up") {
        if(this.y < this.down_bound_y) this.setPosition(this.x, this.down_bound_y);
        if(this.y > this.up_bound_y) this.setPosition(this.x, this.up_bound_y);
    } else {
        if(this.y < this.down_bound_y) this.setPosition(this.x, this.down_bound_y);
        if(this.y > this.up_bound_y) this.setPosition(this.x, this.up_bound_y);
    }
}
```

Fragment 7: Obrada pomeranja i otpuštanja temena

Za potrebe pomeranja i otpuštanja, redefinisano je ponašanje generičkog grafičkog objekta (*Figure*), tako da je nemoguće da se temena nađu van dozvoljenih koordinatnih vrednosti (horizontalno i vertikalno ograničavanje). Pošto (ne)preciznost miša ne treba da utiče na uspešno pomeranje temena fuzzy skupa, pri otpuštanju temena (funkcija *onDragend*), aplikacija koriguje koordinate temena, i vraća ga na najbliže dozvoljene vrednosti. Na taj način, akcija prevlačenja je vrlo fleksibilna i olakšava korišćenje ovog alata.

Pamćenje grafa se realizuje u nekoliko faza. Najpre se instancira *XML request* objekt koji omogućava AJAX pozive od klijenta prema serveru. To znači da se koriste HTTP zahtevi iz *JavaScript*-a, a transfer podataka se vrši u *XML* formatu (Fragment 8).

```
xmlHttp=null;
try{
    xmlHttp=new XMLHttpRequest();// Firefox, Opera 8.0+, Safari
}catch (e){// Internet Explorer
    try{
        xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
    }catch (e) {
        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
}
```

Fragment 8: Uzimanje AJAX XML request objekta

U sledećoj fazi se formira *XML* dokument na osnovu grafika fuzzy termina, a koji treba da se pošalje od klijenta prema serveru (Fragment 9). *XML* dokument je u ovoj fazi string koji se komponuje od *HTTP* zahteva i *XML tag*-ova sa podacima. Podaci se dobijaju iz aktivnog dokumenta Web čitača (radna površina za crtanje). Da bi se dobili fuzzy skupovi (termini), poziva se generička metoda dokumenta – *getFigures*.

```
var xml_data="ajax_request_acceptor.jsp?q=n?xml version=\"1.0\" encoding=\"UTF-8\"?>";
var document=this.palette.workflow.getDocument();
var figures = document.getFigures();
var fterm_started=0; var fvariable_started=0;
for(var i=0;i< figures.length; i++){
    var figure = figures[i];
    if(figure.fuzzysset type!=null && figure.fuzzy term !=null){
        if(figure.fuzzy_variable == null)return;
        else{
            if(fvariable_started == 0){
                xml_data=xml_data+"<f_variable name=\""+figure.fuzzy_variable+"\">";
                fvariable_started = 1;
            }
        }
        if(figure.type == "X1" && fterm_started == 0){
            fterm_started=1;
            xml_data=xml_data+"<fterm type=\""+figure.fuzzysset type+
                "\" name=\""+figure.fuzzv_term+"\"/>";
            xml_data=xml_data+"<x1_coord x=\""+(Math.round(figure.getX()/10)-25)+
                "\" y=\""+(200-figure.getY())/100+"\"/>";
        }
        ...
    }
}
```

Fragment 9: Transformacija grafičkih objekata u XML dokument

Ova metoda vraća niz grafičkih objekata koji se nalaze na radnoj površini. U cikličnoj strukturi se najpre ispituje tip objekta. Ako je objekat temenog tipa, iz njega se izdvajaju

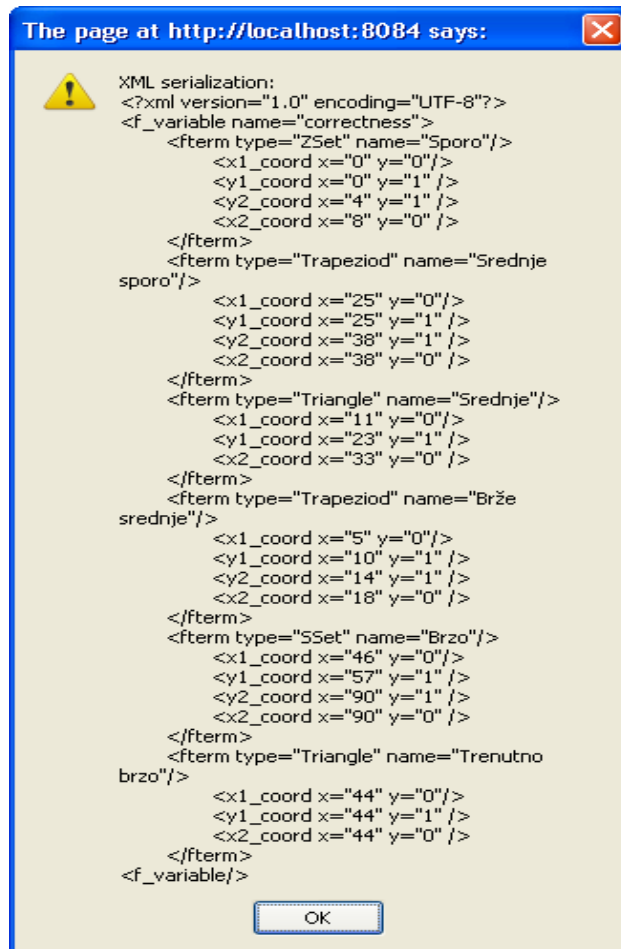
podaci potrebni za transfer. Ovi podaci se konkateniraju (dodaju) na postojeći *XML* string. Iz kodnog fragmenta se može zaključiti da se konektori (stranice) temena ne serijalizuju. Njihovo postojanje je podrazumevajuće.

Na kraju, transfer podataka se ostvaruje proceduralnim pozivanjem metoda *XML request* objekta (Fragment 10).

```
xmlHttp.open("POST",xml_data,true);
xmlHttp.onreadystatechange = callback;
xmlHttp.send(null);
```

Fragment 10: Transfer podataka od klijenta ka serveru korišćenjem AJAX poziva

AJAX HTTP request objekat otvara konekciju prema serveru, podešava je za *POST* režim (podaci koji se šalju nisu transparentni), i kroz nju šalje XML dokument (*xml_data*). Stanje objekta (transfera) se menja, i prati korišćenjem AJAX *callback* mehanizma. Preko ovog parametra, server šalje kodove 0 – 4 (npr. u trenutku instanciranja stanje je 0, ako je konekcija uspešno otvorena, stanje se menja u 1, kada je transfer završen, prelazi se u konačno stanje 4).



Ilustracija 93: Podaci dijagrama (slike X) serijalizovani u XML formatu

Na prethodnoj ilustraciji je prikazan XML dokument koji predstavlja struktuiranu tekstualnu reprezentaciju kreiranog dijagrama termina (Ilustracija 94). Radi optimizacije

sadržaja koji se prenosi mrežom, iskorišćeni su principi vektorizovane grafike, prenose se samo neophodni podaci koji mogu da se ponovo grafički interpretiraju, a koji su dovoljni za uspešno rezonovanje primenom fuzzy logike. Oni u sebi sadrže konkretne vrednosti za korišćene fuzzy koncepte (poglavlje 5.4.5.1). U korenu dokumenta je *f_variable* tag koji enkapsulira kriterijum za ocenjivanje (koncept *FuzzyVariable*). Ovaj tag sadrži atribut *name* – naziv varijable. Kao što kriterijum sadrži termine, tako *f_variable* tag sadrži podtagove *f_term* (koncept *FuzzyTerm*) koji imaju po dva atributa: naziv – *name* i tip fuzzy skupa (koncept *FuzzySet*) kojima su predstavljeni – *type*. Tagovi *f_term* su takođe struktuirani, radi opisivanja karakterističnih tačaka kojima je određen fuzzy skup – podtagovi *x1_coord*, *x2_coord*, *y1_coord* i *y2_coord*. Ovi tag-ovi (koncept *FuzzyPoint*) imaju po dva atributa *x* i *y* koji predstavljaju karakteristične ulazne vrednosti fuzzy varijable (*x*) i njima pridružene funkcije pripadanja (*y*).

5.4.7.4 Korisnički interfejs za definisanje pravila ocenjivanja

Drugi deo pripreme za napredno ocenjivanje predstavlja definisanje pravila (Ilustracija 95). Da bi se umanjila kompleksnost definisanja pravila, alat automatski generiše tabelarni (matrični) prikaz svih kriterijuma i njihovih termina (*Defining assessment rules*).

The screenshot illustrates the 'Defining assessment rules' interface, which is divided into several panels. The top-left panel, titled 'Ekspertsko ocenjivanje: odabir kriterijuma', shows a list of criteria: Brzina, Tacnost, Netacnost, and Rezultat. The 'Brzina' criterion is selected. A green callout 'Selecting criteria' points to this panel. The top-right panel, titled 'Brzina (procenat utrošenog vremena)', shows the definition of terms for 'Brzina'. It lists 'fast', 'medium', and 'low' with their respective percentage intervals. A yellow callout 'Percentage of students which belongs the term interval' points to this panel. The middle-left panel, titled 'Ekspertsko ocenjivanje: rezultati', shows a table of inferred results. A yellow callout 'Inferred results' points to this panel. The middle-right panel, titled 'Tacnost', shows the definition of terms for 'Tacnost'. A green callout 'Defining terms' points to this panel. The bottom-left panel, titled 'Ekspertsko ocenjivanje: rezultati', shows a table of inferred results. A yellow callout 'Inferred results' points to this panel. The bottom-right panel, titled 'Rezultat', shows the definition of terms for 'Rezultat'. A green callout 'Defining terms' points to this panel.

Defining assessment rules

	Tacnost	Brzina	Netacnost	Rezultat
1	low	high	low	uns. tisfied
2	low	medium	low	unsatisfied
3	low	slow	low	unsatisfied
4	medium	high	low	good
5	medium	medium	low	satisfied
6	medium	slow	low	satisfied
7	high	high	low	excellent
8	high	medium	low	excellent
9	high	slow	low	good

Ekspertsko ocenjivanje: rezultati

Indeks	Ime	Prezime	Opisna
121/2004	Mapro	Живкович	satisfied
20/2004	Минах	Личих	excellent
178/2004	Јована	Марић	good

Ilustracija 94: Definisanje pravila ocenjivanja

Na taj način se izbegava da se greškom definiše više pravila za iste vrednosti (termine) kriterijuma uključenih u ocenjivanje, ili još gore, da se propusti definicija pravila za neke vrednosti ulaznih termina (čime bi sistem mogao da se nađe u nedefinisanoj stanju). Nastavnik može da se u ovoj fazi vrati korak nazad i redefiniše termine kriterijuma (paneli označeni kao *Selecting criteria* i *Defining terms*). U tom slučaju, alat odmah

obezbeđuje i podatak o procentualnom broju studenata koji pripadaju redefinisanom terminu.

U tabeli pravila, nastavnik samo treba da izabere za svaku automatski generisanu kombinaciju ulaznih termina (na slici su predstavljeni *tačnost*, *brzina* i *netačnost*), jedan od definisanih termina izlaznog kriterijuma (*Rezultat*). Nakon toga alat je spreman da obavi napredno ocenjivanje studenata. Rezultati se dobijaju u posebnoj panelu (*Inferred results*).

5.4.7.5 Implementacioni detalji fuzzy rezonovanja

Kao što je prethodno opisano, fuzzy rezonovanje se odvija u fazama (poglavlja 5.4.3.3, do 5.4.3.6). U prvoj fazi (*fazifikacija*) vrši se transformacija diskretnih ulaznih vrednosti u fuzzy vrednosti (Fragment 11). Predstavljen je fragment koda funkcije *fuzzyfy* klase *Fuzzyfier* (poglavlje 5.4.6.1), koji prikazuje fazifikaciju vremena koje student provede rešavajući proveru (kod uokviren crvenom linijom). Aplikacija prvo kreira *singleton* fuzzy skup. Zatim se instancira nova fuzzy vrednost (klasa *FuzzyValue*) povezivanjem fuzzy skupa sa odgovarajućom varijablom (klasa *FuzzyVariable*). Time je fazifikacija diskretne ulazne vrednosti završena. Nova fuzzy vrednost se dodaje u kontejner (vektor *values*) koji se prosleđuje na dalju obradu (fazu zaključivanja). Bitno je uočiti i *switch case* uslovno razgranavanje, koje omogućava pridruživanje ulaznih vrednosti odgovarajućim fuzzy varijablama (ispitivanjem tipa varijable). Ovim se rešava filtriranje informacionog šuma pre fuzzy rezonovanja: samo korisnički podaci koji odgovaraju ulaznim kriterijumima su uključeni u rezonovanje.

```

33     for(int i=0;i<variables.size();i++){
34         MTFuzzyVariable mtfv=variables.elementAt(i);
35         switch(mtfv.getFv_id()){
36             case 1://rate
37                 if(student.getDuration()>=0){
38                     FuzzyValue tmp_fvalue = new FuzzyValue(mtfv.getFuzzyVariable(),
39                     new SingletonFuzzySet(student.getDuration()));
40                     values.add(tmp_fvalue);
41                 }
42                 break;

```

Fragment 11: Fazifikacija diskretnih ulaznih vrednosti

U sledećoj fazi vrši se priprema pravila za rezonovanje i punjenje pravila fazifikovanim ulaznim vrednostima. Priprema pravila se realizuje u klasi *RulePreparator*, pomoću dva funkcionalna poziva. Najpre se postavlja egzekutor pravila (metoda *setRuleExecutor*). U sistemu su implementirane tri vrste egzekutora: *Larsen proizvod*, *Mamdani minimum* i *Tsukamoto* (*Sugeno*). Ako se eksplicitno ne promeni, podrazumevani egzekutor je *Mamdani minimum*. Proveravanjem na stvarnim i simuliranim podacima, ovaj tip rezonovanja je davao statistički najpribližnije rezultate nastavničkim (poglavlje 8).

```

for (Enumeration e = this.ht_fuzzy_rules.elements(); e.hasMoreElements() ;) {
    FuzzyRule fr=(FuzzyRule)e.nextElement();
    fr.setRuleExecutor(fr_executor);
    fr.setAntecedentCombineOperator(ac_opreator);
}

```

Fragment 12 Podešavanje pravila pre izvršavanja

Drugi funkcionalni poziv u pripremi pravila je podešavanje operatora kombinovanja ulaznih podataka (metoda *setAntecedentCombineOperator*). Podrazumevajući operator kombinovanja ulaznih fuzzy vrednosti je *mimimum* operator, koji daje kao rezultat najmanju fazifikovanu ulaznu vrednost.

Ulazne vrednosti se mogu množiti – *product* operator. Ovaj operator kao rezultantu daje proizvod skupa od N fazifikovanih ulaznih vrednosti $u(i)$ (izraz 19). Pošto je $u(i)$ zapravo fuzzy skup, i pošto se u kombinovanju razmatraju funkcije pripadanja, vrednost rezultante je kod ovog operatora manja od najmanje ulazne vrednosti.

$$u = \prod_{i=1}^N u(i) \quad (18)$$

Treći tip operatora ulaznih vrednosti je *kompensirano i* (*compensatory and*) [112], koji daje veću vrednost od *mimimum* i *product* operatora, korišćenjem faktora γ u eksponentu izraza (izraz 18). Ako je faktor $\gamma=0$, *kompensirano i* se svodi na *product* operator. Međutim za $\gamma>0$ predstavljen izraz kompenzuje umanjeње rezultante.

$$u = \left(\prod_{i=1}^N u(i) \right)^{1-\gamma} * \left(1 - \prod_{i=1}^N u(i) \right)^{\gamma} \quad (19)$$

Nakon podešavanja, fuzzy pravilo se puni fazifikovanim ulaznim vrednostima (Fragment 13). Pre punjenja, vrši se *reset* pravila – uklanjaju se ulazne vrednosti koje su bile korišćene u prethodnom zaključivanju. Na primer, ako se obavlja napredno ocenjivanje više studenata u jednoj sesiji, potrebno je ukloniti podatke jednog studenta, pre nego što se u premisu pravila dodaju podaci sledećeg studenta (plavo uokviren kod).

```
for (Enumeration e = this.ht_fuzzy_rules.elements(); e.hasMoreElements();) {
    MTFuzzyRule fr=(MTFuzzyRule)e.nextElement();
    fr.removeAllInputs();
    for(int i=0;i<v_fuzzy_values.size();i++){
        FuzzyValue tmp_fvalue=(FuzzyValue)v_fuzzy_values.elementAt(i);
        fr.addInput(tmp_fvalue);
    }
}
```

Fragment 13: Punjenje pravila ulaznim vrednostima

U svako pravilo se dodaju sve fazifikovane ulazne vrednosti (crveno uokviren kod). Specifičnim dizajnom korisničkog interfejsa (poglavlje 5.4.7.5), nastavnici su primorani da definišu pravila za svaku kombinaciju vrednosti nezavisnih varijabli (ulaznih kriterijuma). Ne postoje suvišne ulazne vrednosti, niti varijable koje se ne razmatraju u pravilima. Ovaj sistem za fuzzy rezonovanje je optimizovan na različitim nivoima, a pouzdan rad je omogućen jakom spregom između fuzzy pravila, fuzzy varijabli i fuzzy vrednosti u dizajnu baze znanja.

Nakon dodavanja ulaznih vrednosti, fuzzy pravilo je spremno za korišćenje (Fragment 14). U istoj iterativnoj strukturi u kojoj se dodaju ulazne vrednosti, može se obaviti i izvršenje pravila. Izvršavaju se samo pravila kod kojih postoji poklapanje između uslova

datih u premisi i ulaznih vrednosti (metoda *testRuleMatching*). Ovo poklapanje najčešće nije potpuno (100%). Provera poklapanja nije neophodna za rezonovanje. Ako nema provere, sva pravila u bazi znanja će se izvršiti. Ako je broja pravila veliki, njihovim izvršenjem se formira kompleksan rezultujući fuzzy skup sa neizvesnim rezultatom defazifikacije. Namena provere poklapanja je da se smanji informacioni šum, koji se stvara u navedenom slučaju. Za merenje poklapanja koristi se mehanizam praga (*threshold*), koji se može eksplicitno (najčešće empirijski) podešavati. Promenom praga, kriterijumi za izvršenje pravila se ublažavaju, ili zaoštravaju.

```
if (fr.testRuleMatching()) {
    fvv = fr.execute(); //izvrši pravilo
    result_fvalue = fvv.fuzzyValueAt(0);
    if (globalOutput == null) { //ako je prazna
        globalOutput = result_fvalue;
    } else { //ako nije prazna dodaj u skup
        globalOutput = globalOutput.fuzzyUnion(result_fvalue);
    }
}
```

Fragment 14: Izvršenje pravila (zaključivanje)

Kao što je napomenuto, izvršenje pravila (metoda *execute*) rezultuje novom fuzzy vrednošću (fuzzy skupom). Ova vrednost se dodaje na globalnu rezultujuću vrednost (*globalOutput*) primenom neke od operacija. Najčešće se koriste operacije na skupovima (unija, presek, komplement), ali se mogu izvršiti i specifični zahvati kao što su *Sum* operacija, koja sabira vrednosti funkcije pripadanja rezultujućih fuzzy skupova (posledica sabiranja je da funkcija pripadanja ima i vrednosti veće od 1, tako da se na rezultujući skup treba primeniti normalizacija), ili α presek, koji filtrira skup samo na one vrednosti zavisne varijable koje su ne manje od zadatog kriterijuma α .

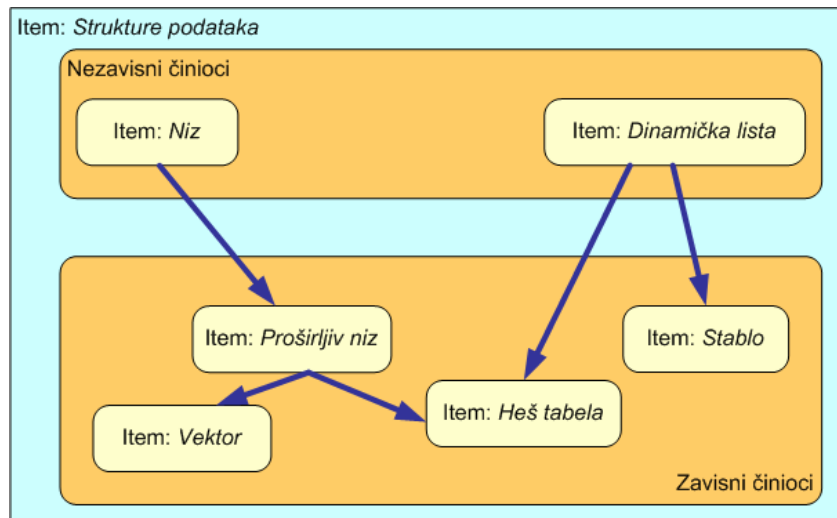
Pošto se sva izabrana (ili ne) pravila izvrše, dobijen je globalnu rezultujući fuzzy skup (*globalOutput*). Da bi rezultat rezonovanja bio primenjiv, poziva se neka od metoda defazifikacije: *momentDefuzzify* (centroid), *centerOfAreaDefuzzify*, *weightedAverageDefuzzify* (metod težinske sredine), *maximumDefuzzify* (poglavlje 5.4.3.6). Svaka od navedenih metoda transformiše rezultujuću fuzzy vrednosti u diskretnu vrednost – realan broj, čime se proces fuzzy rezonovanja završava.

5.5 Definisane relacije između predmetnih činilaca

U svim aktuelnim standardima (poglavlje 3.2) za organizaciju sadržaja podrazumeva se mogućnost kreiranja relacija između predmetnih činilaca (*items* – poglavlje). Pored hijerarhijske (vertikalne) organizacije, postoji mogućnost formiranja poprečnih veza između činilaca na istom nivou strukture (Ilustracija 96). Na primeru su prikazane relacije pod-činilaca u činiocu *Strukture podataka*. Relacija između *Strukture podataka* i njenih pod-činilaca je podrazumevana agregacija (*has_a* veza). Pod-činioци mogu da budu *nezavisni* i *zavisni*. Nezavisni činioци su oni za koje nema posebnih preduslova definisanih za isti hijerarhijski nivo da bi im student pristupio. Ovi činioци su inicijalno uvek vidljivi u navigaciji po sadržaju nad činioца.

Zavisni činioци su povezani za nezavisne relacijama tipa *se zasniva na* (*is based on*). Inverzna relacija *je osnova za* (*is basis for*) je implicitna (ne treba se eksplicitno

definisati). U sistemu se definiše najviše jedna relacija između dva određena činioca. Jedan činilac može da ima relacije sa više drugih činilaca istog nivoa hijerarhije.



Ilustracija 95: Poprečne relacije između činilaca

Činioci i relacije obrazuju usmereni graf u kome su činioci čvorovi, a relacije lukovi. Ovaj graf ne sme da bude cikličan. Navedena pravila u postavljanju relacija olakšavaju njihovo održavanje u smislu izbegavanja ciklične zavisnosti. Na prethodnoj ilustraciji predstavljene su relacije pod-činalaca (npr. lekcija, ili nastavnih jedinica) u činiocu (npr. tematskoj celini, ili poglavlju) *Struktura podataka*. Postoje dva nezavisna (bezuslovna) činioca – *Niz* i *Dinamička lista*. Ako student pristupi sadržajima *Struktura podataka* ova dva činioca će uvek da budu vidljiva. Svi ostali su zavisni činioci. Na primer, ako se savladaju (nauče) nizovi, student može da pristupi povezanom činiocu *Proširljiv niz*. Činiocu *Vektor* student može da pristupi ako savlada *Proširljiv niz*. Student može da pristupi i činiocu *Heš tabela* samo ako je, pored proširljivih nizova, prethodno savladao i sadržaje činioca *Dinamička lista*.

Relacije, zajedno sa zadacima i pitanjima za proveravanje znanja iz konkretnih činilaca, omogućavaju rešavanje adaptivnosti navigacije i sadržaja u sistemu. Ako nisu definisani sadržaji za samotestiranje u toku učenja, navigacija postaje potpuno slobodna, a sadržaji se ponašaju statičke stranice materijala – nepromenjive su (poglavlje 6.1.4).

5.5.1 Vizualizator raspoređivanja (*Sequencing Visualizator*)

Pri dizajnu nastavničkog interfejsa, za definisanje relacija između predmetnih činilaca, nastojalo se da isti bude intuitivan i da ne zahteva posebno obučavanje, niti posebnu stručnost u radu računaram.

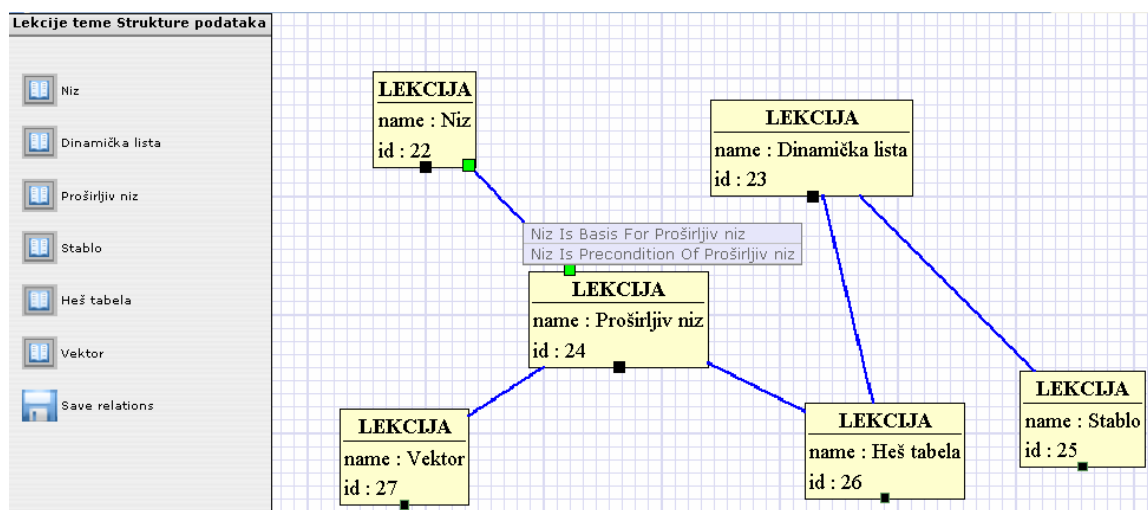
Predmet: Osnove programiranja / Tema: Strukture podataka / Relacije lekcija

Lekcija preduslov	Relacija	Zavisna lekcija	Akcija
Niz	je osnova za	Proširljiv niz	Brisanje
Dinamička lista	je osnova za	Stablo	Brisanje
Dinamička lista	je osnova za	Heš tabela	Brisanje
Proširljiv niz	je osnova za	Heš tabela	Brisanje
Proširljiv niz	je osnova za	Vektor	Brisanje

[Visual editor](#)
[Dodavanje](#)

Ilustracija 96: Osnovni prikaz za dodavanje i uklanjanje relacija između činilaca

Na prethodnoj slici (Ilustracija 97) je prikazan izgled prvobitnog editora relacija između činilaca. Ako je broj činilaca koji treba da se poveže mali, nastavnik lako može da se snađe sa interfejsom koji je zasnovan na tekstualnim komponentama (kontrolama za izbor, unos, linkovima i komandnim dugmadima). Međutim, kompleksnost korišćenja je u direktnoj srazmeri sa brojem činilaca koji treba da se povežu. Iz navedenog razloga kreiran je grafički alat koji bi omogućio nastavniku da dodaje relacije na bazi selekcije, prevlačenja i otpuštanja konektora između grafičkih reprezentacija činilaca (Ilustracija 98).

**Ilustracija 97: Izgled vizuelnog alata za modelovanje relacija između lekcija**

Nakon izbora predmetnog činioca, sistem nudi nastavniku standardan način dodavanja relacija i vizuelni editor relacija. Izborom vizuelnog editora, pojavljuje se radna površina koja na levoj strani sadrži plivajuću paletu sa alatima. Ova paleta se dinamički kreira i sadrži za svaki podčinilac jedno komandno dugme sa čije desne strane se nalazi naziv činioca. Poslednje komandno dugme u nizu je namenjeno pamćenju editovanih relacija.

Ako su relacije prethodno bile kreirane, one će se prikazati na radnoj površini neposredno po pojavljivanju radne površine. Inače nastavnik prvo treba da doda činioce na radnu površinu. Isti činilac se ne može dvaput pojaviti na radnoj površini. Dodavanje se vrši klikom na odgovarajuće dugme, nakon čega se odabrani činilac pojavljuje na radnoj površini, predstavljen kao pravougaonik sa upisanim tipom (*lekcija*), nazivom i identifikatorom. Na sredini donje stranice pravougaonika nalazi se crni kvadratić koji predstavlja priključak za konekcije. Konekcija se dodaje kada se pritisne na priključak, a

zatim prevlači do željenog činioca (njegovog priključka). Činioci lekcija mogu da se raspoređuju (pomeraju), konekcije mogu da se prespajaju, a oba objekta mogu da se obrišu.

5.5.2 Implementacioni detalji vizuelizacije povezivanja činilaca

Za implementaciju vizualnog alata za dodavanje relacija korišćeni su isti koncepti kao kod interfejsa za definisanje naprednog ocenjivanja (poglavlje 5.4.7.2 i 5.4.7.3). Ovaj alat je aplikacija koja se izvršava na većim delom na strani klijenta, u prozoru Web pretraživača. Implementirana je u skript jeziku (*JavaScript*). Klase dizajnirane za crtanje i klase interfejs kontrola, implementirane su izvođenjem iz *Draw2D* [114] biblioteke. Alat se dinamički kreira, po zahtevu (*http request*) za isporučivanjem poslatim od strane klijenta (Fragment 15).

```
<%= "<script> var argm=\""+LessonsBean.getLessonsToJS()+"\"; "%>
    var workflow = new draw2d.Workflow("paintarea");
    var w = new draw2d.GUIPalette(argm);
    workflow.setToolWindow(w);
    w.setPosition(0,0);
    window.onscroll=function(){workflow.onScroll()};
<%= "</script>" %>
```

Fragment 15: Dinamičko kreiranje skript fragmenata

Pošto se u paleti sa alatima nalaze komandna dugmad pod-činilaca (lekcija) odabranog činioca (teme), skript koji vrši dodavanje ovih kontrola se priprema na osnovu podataka na serverskoj strani (metoda *getLessonsToJS*).

```
package graph_utils;
import java.util.Vector;
import unis_portal.cam_ontology.Item;
public class Figure extends Item{
    public int x;
    public int y;
    public Vector <Connection>targets=new Vector();
    public void Figure(){}
}
```

Fragment 16: Pomoćna klasa *Figure*

Rekonstrukcija grafa na osnovu prethodno unešenih relacija se obavlja korišćenjem klase *Figure* (Fragment 16). Ova klasa predstavlja grafičku reprezentaciju predmetnog činioca, tako da jednostavno proširuje klasu *Item* (poglavlje 5.3.2.2), koordinatama gornjeg levog ugla pravougaonika i vektorom ciljnih konekcija (od posmatrane instance ka ostalim objektima).

```

<%
    for (int i = 0; i < figures.size(); i++) {
        Figure f = figures.elementAt(i);
        creation_script += "var object_" + f.title + " = new draw2d.shape.uml.Class(\"LEKCIJA\");\n" +
            "object_" + f.title + ".addAttribute(\"name\", \"\" + f.title + "\");\n" +
            "object_" + f.title + ".addAttribute(\"id\", \"\" + f.id + "\");\n" +
            "workflow.addFigure(object_" + f.title + ", \"\" + f.x + ", \"\" + f.y + ");\n\n";
    }
%>

```

Fragment 17: Dinamičko dodavanje grafičkih objekata na radnu površinu

Skript koji crta zapamćen graf formira serverska aplikacija, neposredno pre slanja zahtevanog sadržaja (Fragment 17). Preko klase srednjeg sloja aplikacije, dobija se vektor *figures* objekata klase *Figure*. U iteraciji se koristi jedan po jedan objekat, koriste se podaci o nazivu, identifikatoru i koordinatama da bi se kreirao string koji će se interpretirati kao skript na klijentskoj strani i koji će da proizvede pri tome crtanje objekata.

```

<%
    for (int i = 0; i < figures.size(); i++) {
        Figure f = figures.elementAt(i);
        for (int j = 0; j < f.targets.size(); j++) {
            Connection _conn = (Connection) f.targets.elementAt(j);
            creation_script += "var con = new draw2d.shape.uml.Connection();\n" +
                "con.setSource(object_" + f.title + ".portBottom);\n" +
                "con.setTarget(object_" + _conn.target + ".portBottom);\n";
            if (_conn.conn_type.equals("is_basis_for")) {
                creation_script += "con.setColor(new draw2d.Color(0,0,255));\n" +
                    "con.relation_type=\"is_basis_for\";\n";
            } else {
                creation_script += "con.setColor(new draw2d.Color(255,0,0));\n" +
                    "con.relation_type=\"is_precondition_of\";\n";
            }
            creation_script += "workflow.addFigure(con);\n";
        }
    }
%>
<%= "" + creation_script %>

```

Fragment 18: Dinamičko dodavanje konekcija između objekata

U sledećoj iteraciji, pre isporuke stranice, serverska aplikacija dodaje skript koji crta konekcije između objekata (Fragment 18). Konekcije se crtaju prostim povezivanjem izvorišnog objekta klase *Figure* sa svim ciljnim objektima, sadržanim u vektoru *targets*. Ovi objekti su takođe enkapsulirani. Pošto, pored naziva ciljnog činioca, treba da se pamti i tip konekcije, kreirana je posebna klasa *Connection* koja u sebi sadrži ove podatke. U JavaScript-u postoji istoimena klasa, kojoj se prosleđuje identifikator izvorišnog i ciljnog objekta. Kada se na klijentskoj strani doda instanca *Connection* na radnu površinu, ako su objekti koje treba da spoji prisutni, i relacija će da bude nacrtana.

```

var document=this.palette.workflow.getDocument();
var figures = document.getFigures();
var id=this.id;
    for(var i=0;i< figures.length; i++){
        var figure = figures[i];
        if(figure.obj_id == id){
            alert("Object already exists in graph");
            this.palette.setActiveTool(null);
            return;
        }
    }

var object = new draw2d.shape.uml.Class(this.type);
object.addAttribute("name",this.name);
object.addAttribute("id",this.id);
this.palette.workflow.getCommandStack().execute(
    new draw2d.CommandAdd(this.palette.workflow,object,x,y));

```

Fragment 19: Dodavanje novog činioca na radnu površinu

Za crtanje novog dijagrama relacija koriste se iste klase, njihovi atributi i pozivaju se iste metode (Fragment 19). Na prethodnom primeru je prikazan kod rukovaoca događajem dodavanja činioca na radnu površinu. Ovaj *handler* se aktivira kada je pritisnuto jedno od dugmadi činilaca na paleti sa alatima. Najpre se proverava da li na dijagramu postoji izabrani činilac (crveno uokviren kod). Ako postoji, aplikacija generiše odgovarajuću poruku (*alert*) korisniku i resetuje stanje palete nakon čega se metoda završava. Inače, se nastavlja sa crtanjem objekta na prethodno opisan način (plavo uokviren kod). Da bi se omogućilo *undo* i *redo* akcije koristi se posredno izvršenje preko klase *CommandStack*-a, koja pamti zadate komande, nova i prethodna stanja. Za crtanje konekcija koriste se klase i funkcionalni pozivi, prethodno opisani u ovom poglavlju.

6 Studentska aplikacija

U narednim poglavljima opisani su najvažniji detalji iz dizajna i implementacije studentske aplikacije, sa akcentom na rešavanje personalizovane adaptacije sadržaja učenja i navigacije kroz prostor učenja.

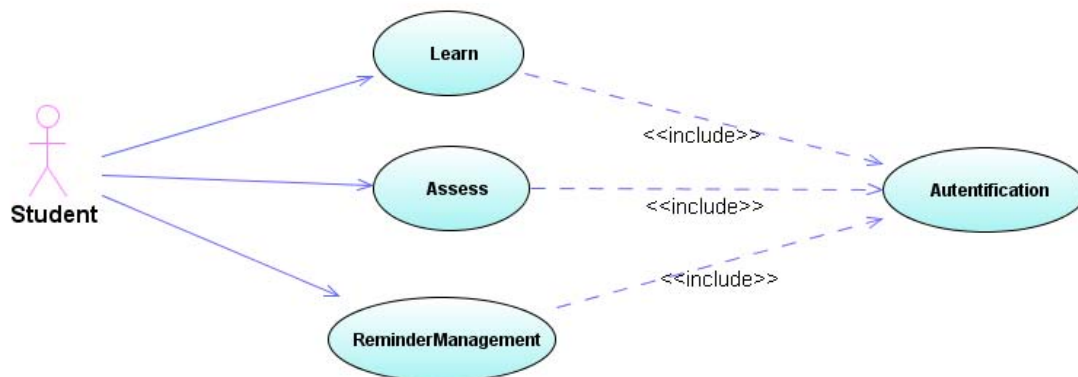
6.1 Dizajn i implementacija

U sledećim paragrafima opisani su u potpunosti slučajevi korišćenja studentske aplikacije, paketi klasa koji ju sačinjavaju i konceptualni dijagrami svakog paketa ponaosob sa težištem na relacijama između koncepata. Dijagramima aktivnosti predstavljeni su različiti aspekti korišćenja sistema. Dijagramima interakcije predstavljeni su detalji implementacije produkcionih pravila u rešavanju adaptivne navigacije i komponovanja sadržaja.

6.1.1 Slučajevi korišćenja

Korišćenje studentske aplikacije može da se prikaže sa tri slučaja korišćenja nultog nivoa (Ilustracija 99):

- Učenje – najvažniji SK, pošto se tiče glavne funkcionalnosti sistema. Na ovom (nultom) nivou, pod učenjem se podrazumeva ne samo saznajni proces, već i svi ostali koncepti u najširem smislu te reči (konstruktivistički, socijalni, emotivni, motivacioni i kognitivni koncepti).
- Proveravanje – u direktnoj sprezi sa učenjem; postoje dva vida proveravanja – samoocenjivanje, koje studenti samostalno obavljaju u toku procesa učenja, kako bi stekli uvid u naučenost materijala, i ispiti (klasifikacije). Rezultate samoocenjivanja mogu da koriste nastavnici kako bi pratili napredovanje studenata i blagovremeno pružili potrebnu pomoć i podsticaj. Rezultati ispita se evidentiraju u zvanični studentski zapis, sačinjavajući njegov portfolio, koji se inicira pri započinjanju školovanja, proširuje i dopunjava rezultatima u toku školovanja, omogućavajući zainteresovanima da u svakom trenutku imaju kompletan profil, sposobnosti, veštine i znanja studenta.
- Upravljanje podsetnikom – kroz ovaj SK studenti mogu sami da organizuju svoje radne obaveze i da koriste pomoć koju im pruža sistem i ostali korisnici (nastavnici i drugi studenti) kroz sistem.
- Autentifikacija – kao i kod nastavničke aplikacije, ovaj SK je neophodan radi autorizovanog pristupa sistemu. U studentskoj aplikaciji postoji i jedan aspekt više - identifikacija studenta je neophodna kao preduslov adaptacije sistema prema potrebama individualnog studenta.



Ilustracija 98: Slučajevi korišćenja 0-tog nivoa studentske aplikacije

Od svih navedenih, najkompleksniji SK je učenje, a najspecifičniji je proveravanje. Učenje i upravljanje podsetnikom može da se odvija u kombinovanom modu (student može pristupiti podsetniku u toku učenja). Za vreme proveravanja, ostale funkcionalnosti sistema su nedostupne, sve do završetka provere.

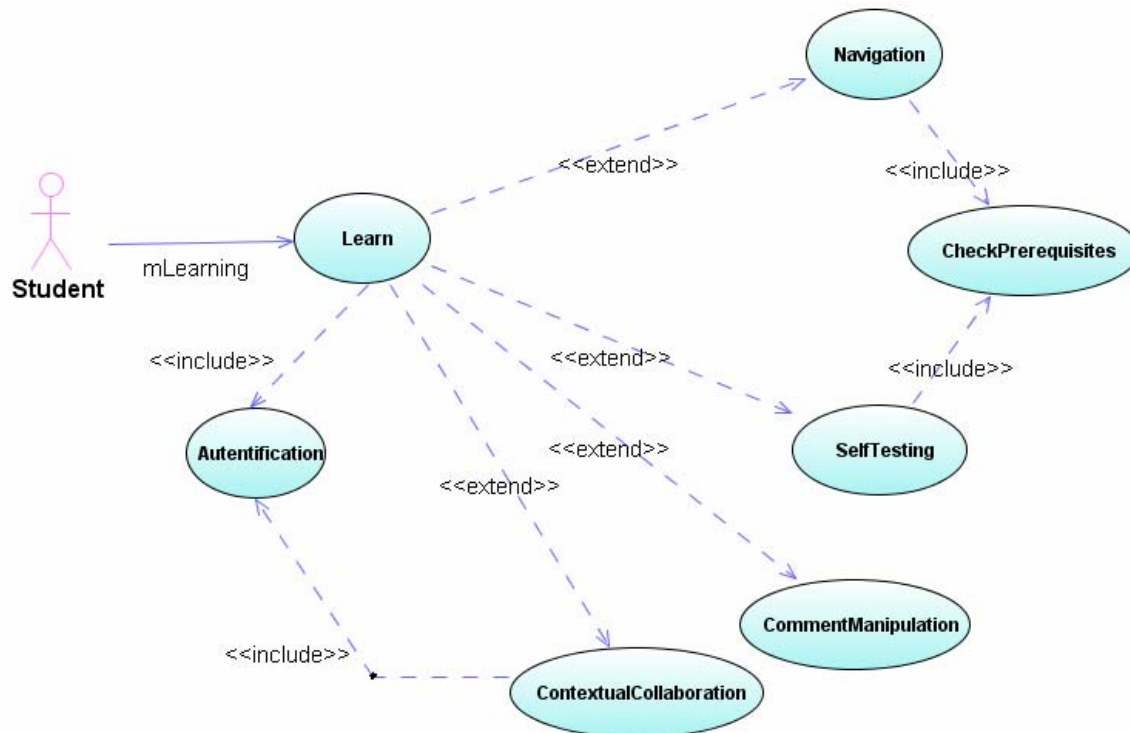
6.1.1.1 Učenje

SK *Učenje* je najkompleksniji, i specijalizuje u pet slučajeva prvog nivoa (Ilustracija 100):

- *Navigation* (Navigacija) – ovaj SK omogućuje studentu da pristupa resursima učenja, da ih koristi na način koji je nastavnik definisao. Resursi su adaptivni tako

da pored predefinisane pedagogije, na njihov sadržaj i način prezentovanja utiče i stanje modela individualnog studenta. U toku ovog SK, sistem prati rad svakog studenta, njegove akcije i ažurira studentski model, uz proveravanje preduslova za pojavljivanje, ili skrivanje resursa.

- *SelfTesting* (Samotestiranje) – za razliku od SK proveravanje, samotestiranje je deo procesa učenja. Ovaj SK omogućava uvid u trenutni nivo znanja studenta, omogućavajući adaptaciju sadržaja i navigacije u prostoru učenja. Studentu samotestiranje omogućava uvid gde treba da učini više napora i na kojim sadržajima da se fokusira da bi, uz istovremenu pomoć sistema, ostvario bolje rezultate.
- *CommentManipulation* (Manipulacija komentarima) – koncept komentara omogućava anotiranje (beleške) sadržaja u toku učenja. Ovaj SK je sličan nastavničkom SK Upravljanje komentarima (poglavlje 5.4.1.2). Ova funkcionalnost je dostupna samo u toku učenja, i povezuje se za konkretan sadržaj. Student može da dodaje komentare, da ih učini vidljivim ostalima, ili da ih ostavi samo za sebe. Isto tako, student može da vidi javne komentare drugih korisnika, za trenutno korišćen sadržaj učenja. Komentari se ne mogu dodavati za resurse samotestiranja.
- *ContextCollaboration* (Kontekstualna saradnja) – kao što se komentari dodaju sadržajima učenja, kontekstualna saradnja predstavlja vid komunikacije između korisnika u izabranom kontekstu učenja. Sistem omogućava studentu da ima uvid u ostale korisnike koji su pristupili zadatom kontekstu učenja (na istoj temi, lekciji, ili materijalu). Ovaj SK omogućava sinhronu i asinhronu komunikaciju. Sinhrona komunikacija je kontekstualni *chat*, a asinhrona je kontekstualni forum.
- *CheckPrerequisites* (Provera preduslova) – ovaj SK je uključen u SK navigacije i samotestiranja. Kada student generiše zahtev za pristupanje nastavnom materijalu, ili samotestiranju, sistem proverava preduslove. Pri tome se obraća studentskom modelu radi dobijanja trenutnog stanja (znanja) studenta iz konkretnog konteksta učenja. Pored provere da li može da pristupi sadržaju, sistem vrši i adaptaciju sadržaja koji se isporučuje. Vršiti se dinamičko komponovanje, u skladu sa statusom studenta.



Ilustracija 99: Studentska aplikacija – SK 1. nivoa za funkcionalnosti učenje

Kao u svim prethodnim SK autentifikacija (provera naloga) korisnika je neophodna. Ovaj slučaj korišćenja se koristi i radi kontrole aktivnosti studenta (npr. pomeranje kursora, korišćenje tastature, skrolovanje teksta). Ako nema znakova aktivnosti u predefinisanim vremenskim periodima, ovaj SK automatski isključuje studenta iz sistema. Studenti na taj način sprečavaju da koriste sistem na nedozvoljeni način, bilo da je to namerno, ili slučajno.

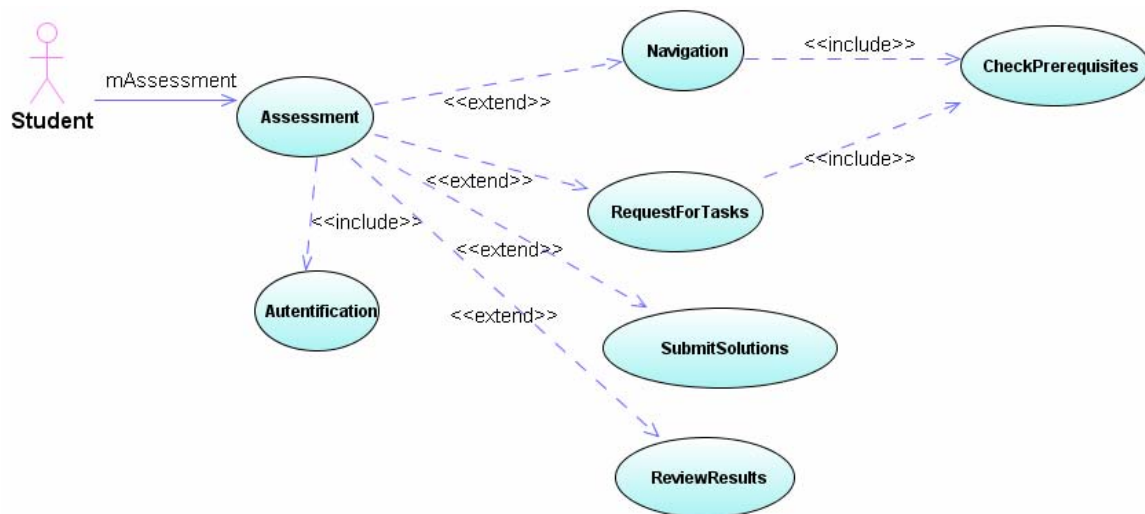
6.1.1.2 Proveravanje

SK proveravanje omogućava primenu različitih tehnika i strategija testiranja. Ovaj SK se sastoji od 4 SK prvog nivoa (Ilustracija 101):

- *Navigation* (Navigacija) – ovaj SK omogućuje studentu kretanje po sadržajima provere. Ono može da bude potpuno ograničeno, potpuno slobodno, ili delimičnim ograničavanjem. Kod ograničene navigacije, sistem vodi studenta postupno kroz zadatke, ne dozvoljavajući mu da promeni tok događanja, navigaciju u nazad, ili bilo kakav oblik selekcije. Ograničena navigacija je pogodna kada se, pored poznavanja gradiva, od studenta traži i brzina odgovora. Potpuno slobodna navigacija omogućava studentu da sam određuje redosled rešavanja zadataka provere. Slobodna navigacija je pogodna kod rešavanja problemskih zadataka, tako da studenti u zavisnosti od koncentracije, mogu da biraju prema težini redosled rešavanja zadataka. Delimično ograničavanje omogućava studentu potpunu i ograničenu navigaciju u toku provere, u zavisnosti od trenutne situacije, statusa studenta i zadatka koji rešava. Delimično ograničena navigacija odgovara proverama u kojima su zadaci različitog tipa, npr ako se uđe u blok pitanja tipa *single/multiple choice* onda ograničavanje, ako se radi

problematski zadatak, onda da se omogući privremeno napuštanje i prelaženje na drugi zadatak. Drugi tip delimičnog ograničavanja je da sistem postupno vodi studenta unapred, ali mu istovremeno dozvoljava navigaciju u nazad.

- *RequestForTasks* (Zahtevanje zadataka) – koristeći ovaj SK, student zahteva od sistema da mu isporuči zadatak (ili zadatke, u zavisnosti od primenjena strategije). Eksplicitno zahtevanje zadataka je moralo da se uvede kao zaseban SK, jer je time uključena mogućnost da sistem studentu u jednom trenutku može da ponudi više provera (u praksi se pokazalo kao korisno kod studenata koji se nalaze u zaostatku sa obavezama, tako da imaju nagomilane provere koje moraju da reše). Ovim SK omogućava izbor konkretne provere, tako da se ostale provere i ostali resursi učenja zaključavaju.
- *SubmitSolutions* (Potvrđivanje rešenja) – ovaj SK omogućuje studentu da eksplicitno potvrdi svoje unose. Time sistem registruje vreme završetka provere, prelazi u fazu ocenjivanja studenskih odgovora i rešenja, isključuje aktivirane resurse i kontrole koje su bile zadejstvovane u fazi proveravanja.
- *ReviewResults* (Pregled rezultata) – ovaj SK omogućuje studentu da pregleda rezultate provere. I u ovom SK su implementirane različite prezentacione tehnike. Na primer, da li se rezultati prikazuju odmah, ili naknadno (npr. nakon uvida nastavnika), da li se prikazuje samo sumarni rezultat, ili sumarni rezultat sa detaljima odgovora i povratnom informacijom. Pregled rezultata se razlikuje i u zavisnosti od primenjenog tipa provere. Ako se radi o proveru kod koje nastavnik pregleda i ocenjuje rešenja i odgovore studenata, rezultati ne mogu da budu odmah dostupni. U ostalim slučajevima potrebna je eksplicitno definisanje tehnike.

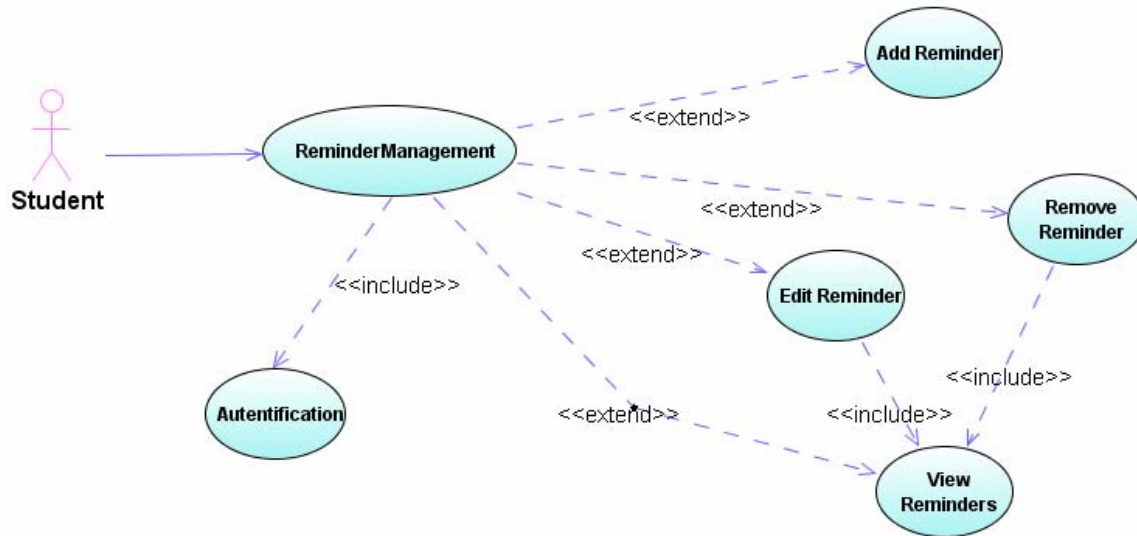


Ilustracija 100: Studentska aplikacija – SK 1. nivoa za funkcionalnosti proveravanja

SK *Navigacija* i *Zahtevanje zadataka* uključuju SK *Proveravanje preduslova*. Na taj način se postiže adaptivnost sistema utoku proveravanja. Ovaj SK omogućava da se zadaci za proveravanje prilagođavaju aktuelnom stanju studenta, i da se student postupno vodi kroz proveru (npr. od lakših ka težim zadacima).

6.1.1.3 Upravljanje podsetnikom

SK Upravljanje podsetnikom (Ilustracija 102) obuhvata funkcionalnosti dodavanja, uklanjanja, pregledanja i editovanja podsetnika. Ova grupa funkcionalnosti omogućava studentu da planira i organizuje svoje obaveze i aktivnosti u procesu učenja.



Ilustracija 101: SK 1.nivoa Upravljanje podsetnikom

Pri pristupu aplikaciji, nakon autentifikacije, sistem proverava obaveze studenta. U slučaju da postoje obaveze koje nisu rešene a nalaze se u planiranom vremenu izvršavanja, sistem šalje indikaciju. Kao i ostali SK, upravljanje podsetnikom je vezano za konkretan kontekst učenja. Međutim podsetnik je uvek aktivan i jedino nije dostupan u toku proveravanja. To znači da će indikacija obaveza da postoji i van konteksta za koji su te obaveze definisane. S druge strane student uvek ima pristup svim svojim obavezama.

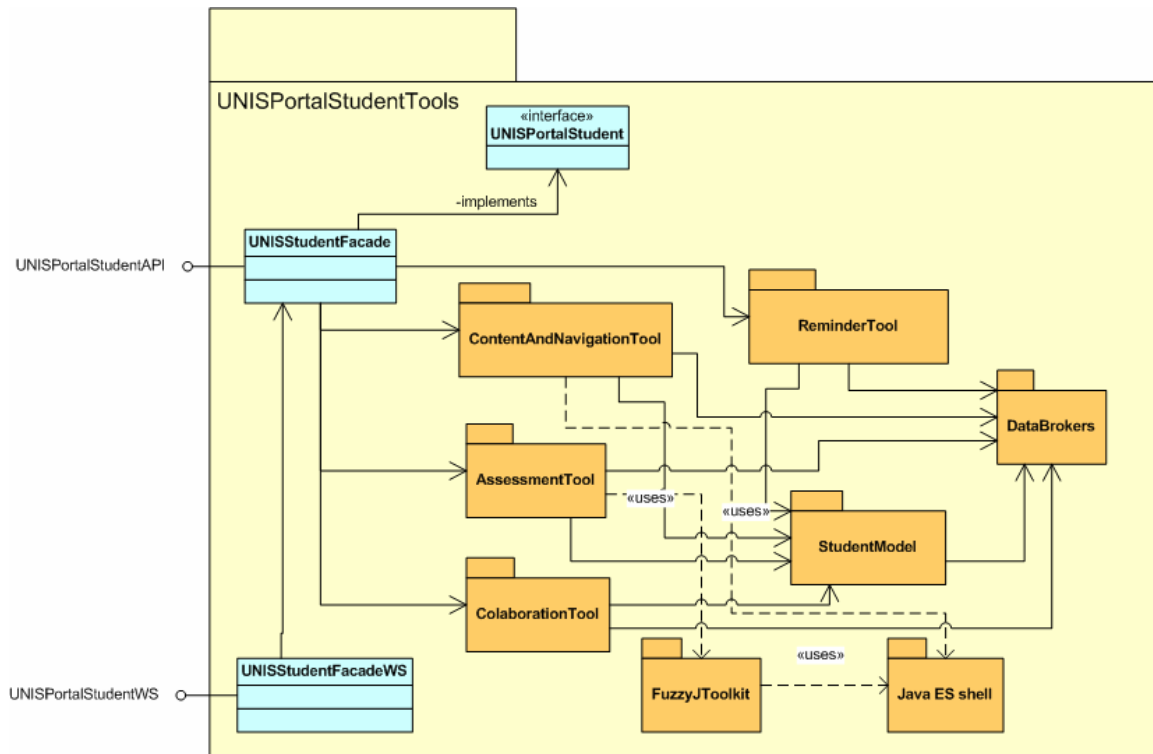
6.1.2 Konceptualni modeli studentske aplikacije

Studentska aplikacija je modularna kao i nastavnička aplikacija (Poglavlje 5.4.2). Moduli su u sistemu predstavljeni paketima klasa, koje su grupisane na osnovu funkcionalnosti. Klase studentske aplikacije su organizovane u dvonivojsku hijerarhiju paketa. Koreni paket je *UNISPortalStudentTools*, koji sadrži pet pod-pakovanja (Ilustracija 103):

- *ContentAndNavigationTool* – paket klasa za korišćenje sadržaja učenja
- *ReminderTool* – paket klasa za rad / upravljanje podsetnikom
- *AssessmentTool* – paket klasa za korišćenje provera
- *CollaborationTool* – paket klasa za komunikaciju između korisnika
- *StudentModel* – paket klasa koje enkapsuliraju model studenta (Poglavlje 3.1)
- *DataBrokers* – paket klasa – posrednika za rad sa podacima

Pored navedenih paketa u korenom paketu su definisani jedan interfejs (*UNISPortalStudent*) i dve klase (*UNISStudentFacade* i *UNISStudentFacadeWS*). U

interfejsu *UNISPortalTeache* su deklarirane metode posredstvom kojih korisnici (druge aplikacije) koriste klase studentskih alata.



Ilustracija 102: Paketi studentske aplikacije

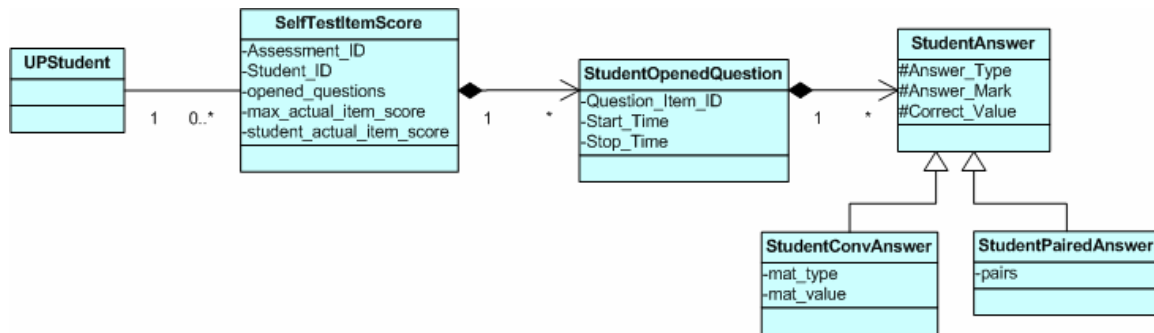
Primenom *Facade* projektnog obrasca [113], dizajnirane su dve klase korenog paketa posredstvom kojih se komunicira izvana sa svim ostalim klasama paketa. Kao kod nastavničke aplikacije, komunikacija se može odvijati na dva načina: konvencionalno (korišćenjem API-ja), ili posredstvom Web servisa (detalji su identični kao u poglavlju 5.4.2).

Pored predstavljenih paketa postoje dva koja predstavljaju platformu za razvoj sistema za zaključivanje. To su shell-ovi za razvoj ekspertskih sistema na bazi predikatske logike (*Java ES Shell*) i na bazi fuzzy logike (*FuzzyJToolkit*). Paket *FuzzyJToolkit* [134] baziran je na paketu *Java ES Shell* (u daljem tekstu *JESS*) [108]. Ove pakete koriste klase namenjene za adaptaciju sadržaja i navigacije (paket *ContentNavigationTool*), i klase namenjene proveravanju znanja studenata (paket *AssessmentTool*).

6.1.2.1 Paket *ContentAndNavigationTool*

Alat za navigaciju i zahtevanje sadržaja je obuhvaćen posebnim paketom – *ContentAndNavigationTool*. Navigaciona i prezentaciona adaptivnost studentske aplikacije je implementirana u klasama ovog paketa. Ovaj paket je najkompleksniji jer, pored adaptivnosti sadržaja učenja, ima ugrađenu funkcionalnost za samotestiranje. Samotestiranje predstavlja sastavni deo procesa učenja i omogućava studentu da nakon proučavanja nastavne celine (npr. lekcije), proveri sopstveno znanje iz iste. Sistem pamti performanse na samotestiranju, ažurira studentski model, i na osnovu toga vrši adaptaciju sadržaja.

Za potrebe samotestiranja dizajniran je konceptualni model (Ilustracija 104). Za svaki činilac predmeta nastavnik može da definiše proizvoljan broj pitanja koja su namenjena samotestiranju (detaljnije u poglavlju „Paket *AssessmentTool*“). U toku učenja student sukcesivno pristupa pitanjima, a sistem nakon svakog samotestiranja ažurira kumulativni rezultat. Ovi podaci su namenjeni za statističku obradu i predstavljaju aktuelno studentsko znanje iz određenog nastavnog činioca (npr. oblasti, ili lekcije). Klasa *StudentItemScore* enkapsulira ove podatke, koji su izvedeni, ali se eksplicitno čuvaju jer nastavnici mogu da modifikuju pitanja, njihov broj i način vrednovanja odgovora. Zahvaljujući klasi *StudentItemScore* ne gubi se semantika, niti konzistentnost studentskih rezultata.



Ilustracija 103: Koncepti (klase) za samotestiranje

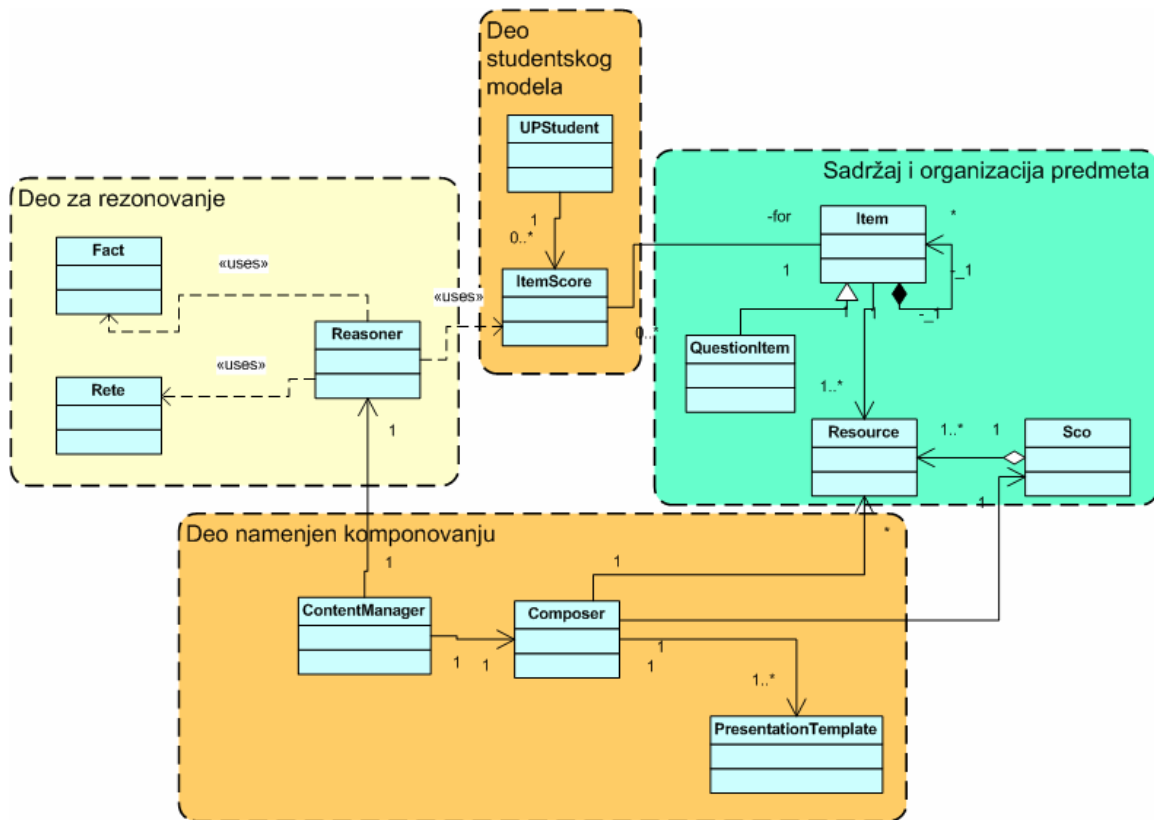
Kumulativni rezultat svakog nastavnog činioca računa se na osnovu svih pitanja kojima je student pristupao u toku ukupnog izučavanja činioca (drugim rečima, iz jedne, ili više sesija). Klasa *StudentOpenedQuestion* čuva ID pitanja i vreme početka i završetka samotestiranja. Na ovaj način se čuva tačan datum i vreme rada, i trajanje kao izveden podatak.

Kao što je klasa *StudentOpenedQuestion* pandam klasi *QuestionItem*, tako je klasa *StudentAnswer* pandam klasi *AnswerItem*. Student na jedno pitanje može da da jedan ili više odgovora. Prema tipovima pitanja, odgovori mogu da budu specifični, tako da klasa *StudentAnswer* specijalizuje u dve izvedene klase:

- *StudentConvAnswer* – klasa koja specijalizuje odgovore na bazi slobodnog unosa teksta, ili odgovore jednostrukog (*single choice*) i višestrukog izbora (*multiple choice*).
- *StudentPairedAnswer* – klasa koja specijalizuje odgovore na zadatak koji traži da se elementi povežu (upare), ili da se elementi prevlače na odgovarajuće odredište.

Nad-klasa *StudentAnswer* pored tipa odgovora, sadrži ocenu odgovora i njegovu korektnu vrednost. Na taj način, u slučaju da dođe do promene u sadržaju odgovora (brisanje ili editovanje objekata klase *AnswerItem*), studentski odgovori i dalje mogu da se analiziraju na osnovu podataka pohranjenih u navedenim klasama studentskih odgovora.

Drugi deo paketa odnosi se na adaptacionu funkcionalnost. Na sledećoj ilustraciji (Ilustracija 105), prezentovani su osnovni koncepti za razumevanje načina na koji se ova funkcionalnost ostvaruje.



Ilustracija 104: Konceptualni model za adaptaciju sadržaja i navigacije

Adaptacija sadržaja vrši se po zahtevu korisnika sistema (klijentske aplikacije). Klasa kojoj se korisnik obraća nazvana je *ContentManager*. Ova klasa, po dobijanju zahteva, proverava preduslove za dobijanje traženog sadržaja. *ContentManager* koristi klasu *Reasoner*, koja obavlja zaključivanje o studentu – njegovom aktuelnom stanju. Klasa *Reasoner* pri tome koristi klase *Rete* i *Fact* iz paketa *Java ES Shell*, i klasu *ItemScore* iz paketa *StudentModel* kako bi proverila da li student ispunjava potrebne preduslove. Zaključivanje se obavlja korišćenjem skupa pravila na čije ulaze se dovode podaci iz *ItemScore* instanci. Pravila se nalaze van objektnog modela, smeštena u bazu znanja, zapamćena u formi skript datoteke, i napisana u CLIPS formatu [135] (detaljnije u poglavlju “Implementacija rezonovanja na bazi pravila”).

Zaključci procesa rezonovanja se dalje koriste kao odluke za dinamičko komponovanje celokupnih sadržaja. Klasa *ContentManager* ovu aktivnost prepušta instanci klase *Composer*, koja koristi specifični prezentacioni šablon i resurse povezane za konkretan činilac (klase *Resource* i *SCO*).

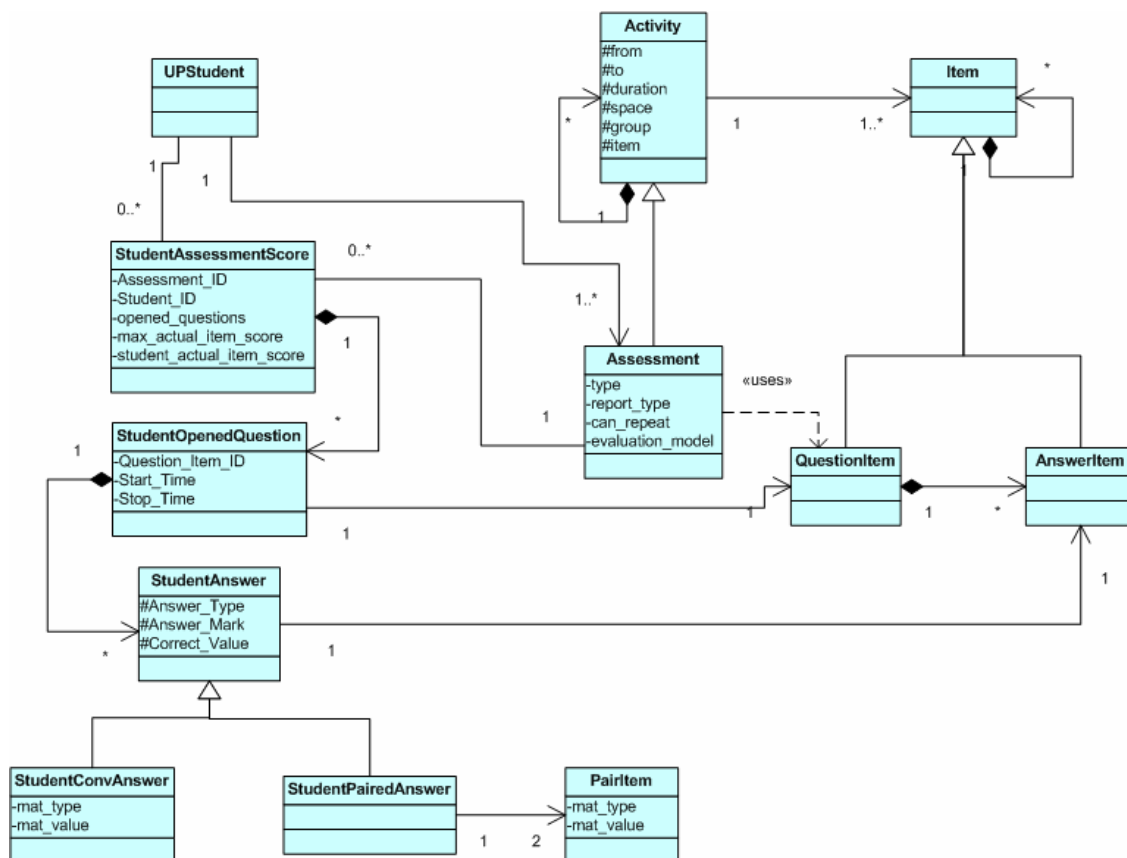
6.1.2.2 Paket *AssessmentTool*

Paket *AssessmentTool* omogućava studentu da pristupi klasifikacionim proverama. Model podataka za potrebe proveravanja predstavlja proširenje modela podataka samotestiranja. Suštinski su uvedena dva nova koncepta:

- Aktivnost (*Activity*) – ovaj globalni koncept enkapsulira bilo kakvu aktivnost koja je definisana u procesu učenja. Uvođenjem ove klase, omogućeno je da se

preciznije definišu aktivnosti u sistemu. Aktivnosti definiše nastavnik. Pri tome se definišu predmet, nastavni činilac(i), nastavna grupa, mesto, vreme i trajanje. Aktivnosti su vezane za činioce (*Item*). Kao što činioči mogu da sadrže podčinioce (npr. teme mogu da sadrže više lekcija), tako i aktivnosti mogu da predstavljaju agregate. Pored navedenog, aktivnosti omogućavaju novi aspekt izveštavanja i prema nastavnicima i prema studentima. Na primer, praćenje napredovanja studenata može da se posmatra iz ugla praćenja njegovih aktivnosti.

- Proveravanje (*Assessment*) - ovaj koncept predstavlja konkretnu aktivnost, koja koristi resurse pitanja, definisane u okviru činilaca. Student može da pristupa jednoj, ili više provera. Za svaku proveru, sistem računa konačni rezultat (*StudentAssessmentScore*), na isti način kao što to čini kod samotestiranja (poglavlje 6.1.2.1).



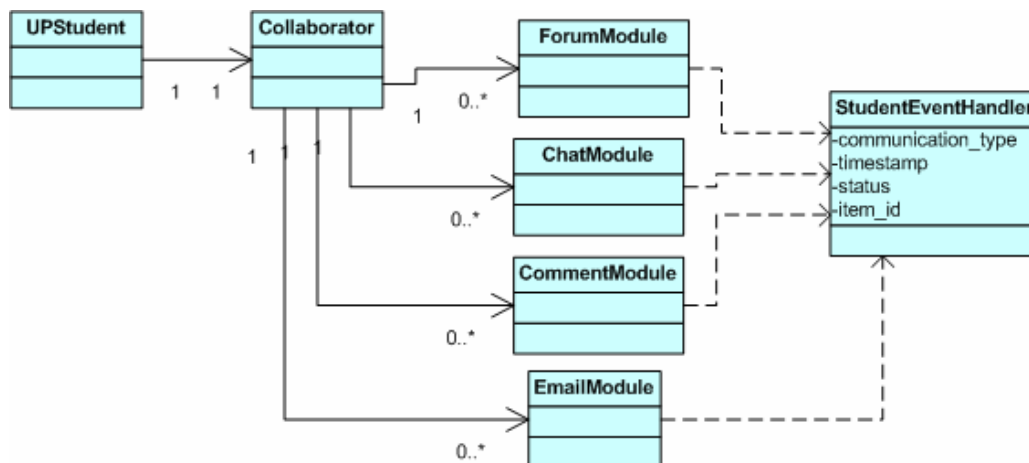
Ilustracija 105: Model podataka za provere

Na predstavljenom konceptualnom modelu (Ilustracija 106), klasa koja ima centralnu važnosti je *Assessment*. Ona predstavlja aktivnost, koja pored nasleđenih atributa, sadrži podatke o tipu provere, načinu izveštavanja prema studentu, mogućnosti ponavljanja provere i načinu ocenjivanja (Poglavlje 6.1.4). Na osnovu ovih parametara i raspoloživih pitanja, sistem kreira sadržaj proveravanja za svakog studenta posebno (Poglavlje 6.1.4).

6.1.2.3 Paket *CollaborationTool*

Paket *CollaborationTool* ima gotovo identične funkcionalnosti kao istoimeni paket nastavničke aplikacije. Studentu je omogućena sinhrona i asinhrona komunikacija sa ostalim korisnicima sistema. Komunikacija se odvija posredstvom klase *Coordinator* (*Facade* UML projektni obrazac [113]). Kao kod nastavničke aplikacije, postoje četiri osnovna komunikaciona modula:

- ForumModule – asinhrona komunikacija na osnovu zadatih ili predloženih tematskih sadržaja.
- ChatModule – sinhrona komunikacija razmenom tekstualnih poruka
- EmailModule – modul namenjen asinhronoj razmeni poruka u vidu elektronske pošte.
- CommentModule – modul namenjen anotaciji sadržaja.

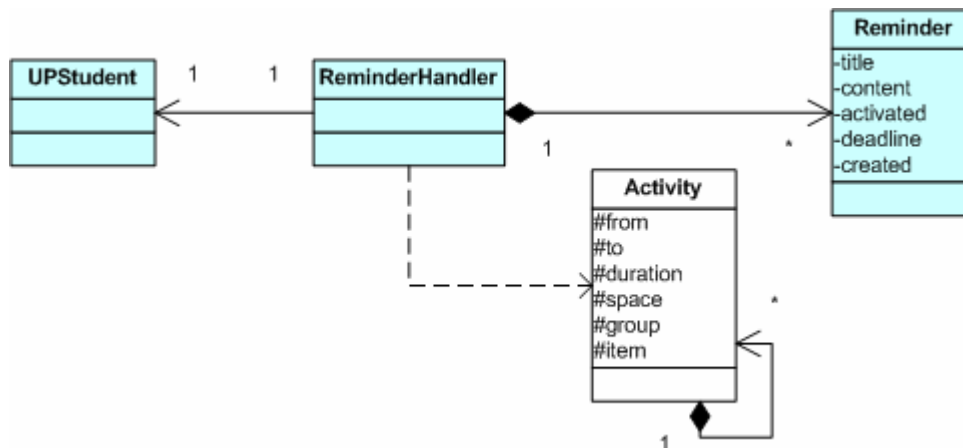


Ilustracija 106: Konceptualni model za kolaboraciju

Za razliku od nastavničkog komunikacionog modela, u studentskom se vrši beleženje aktivnosti. Za tu namenu je dizajnirana klasa *StudentEventHandler*. Svi komunikacioni moduli koriste ovu klasu za pamćenje studentskih logova. Podaci koji se pamte su tip komunikacije, vreme, status i kontekst u kome je akcija izvedena. To znači da se komunikacija u sistemu ne može realizovati van konteksta učenja. Da bi student koristio resurse, on mora da pristupi nastavnom činiocu (npr. temi, ili lekciji), koji predstavlja komunikacioni kontekst. To znači da student, iako koristi komunikacione alate, ne gubi fokus na učenju. Sa druge strane, sistem može da filtrira prikaze aktivnih korisnika, svodeći ih na one koji se nalaze u istom kontekstu. Da bi se izbegla zamka preteranog ograničavanja, i da studenti ne bi koristili kontekstualnu komunikaciju za razmenu poruka van konteksta, sistem dozvoljava i vankontekstualni pristup komunikacionim modulima. Tada se akcije ne registruju, jer iz aspekta učenja, one predstavljaju samo izvor informacionog šuma.

6.1.2.4 Paket *ReminderTool*

Paket *ReminderTool* omogućava studentu da upravlja vlastitim podsetnikom. Takođe posredstvom ovog paketa student ima uvid u sve obaveze zadate od nastavnika.



Ilustracija 107: Konceptualni model podsetnika

Konceptualni model podsetnika je jednostavan (Ilustracija 108). U paketu postoji klasa koja predstavlja interfejs prema ostalim paketima (korisnicima) podsetnika (*ReminderHandler*). Istovremeno ova klasa ostvaruje vezu sa nastavnim aktivnostima studenta (klasa *Activity*). Podaci za podsećanje su enkapsulirani u klasu *Reminder*. Pri korišćenju studentske aplikacije, automatski se instancira jedan objekat klase *ReminderHandler*, koja dalje vrši proveru podataka podsetnika. Ako je podsetnik aktiviran, biće prikazan korisniku. Isto tako objekat *ReminderHandler*-a proverava u aktivnostima definisanim za grupu kojoj pripada student, da li ima obaveza. Za svaku obavezu, čiji podatak *od (from)* je stariji od aktuelnog sistemskog vremena, kreira se podsetnik (ako prethodno nije kreiran), koji je odmah aktivan. Podatak *do (to)* iz aktivnosti se mapira u podatak *rok (deadline)* podsetnika. U podatku *sadržaj (content)* podsetnika se zapisuje referenca na obrađenu aktivnost. U slučaju isticanja roka podsetnika, handler obaveštava sistem, tako da aplikacija prikazuje upozorenje korisniku, u vidu prioritete poruke.

6.1.2.5 Paket *StudentModel*

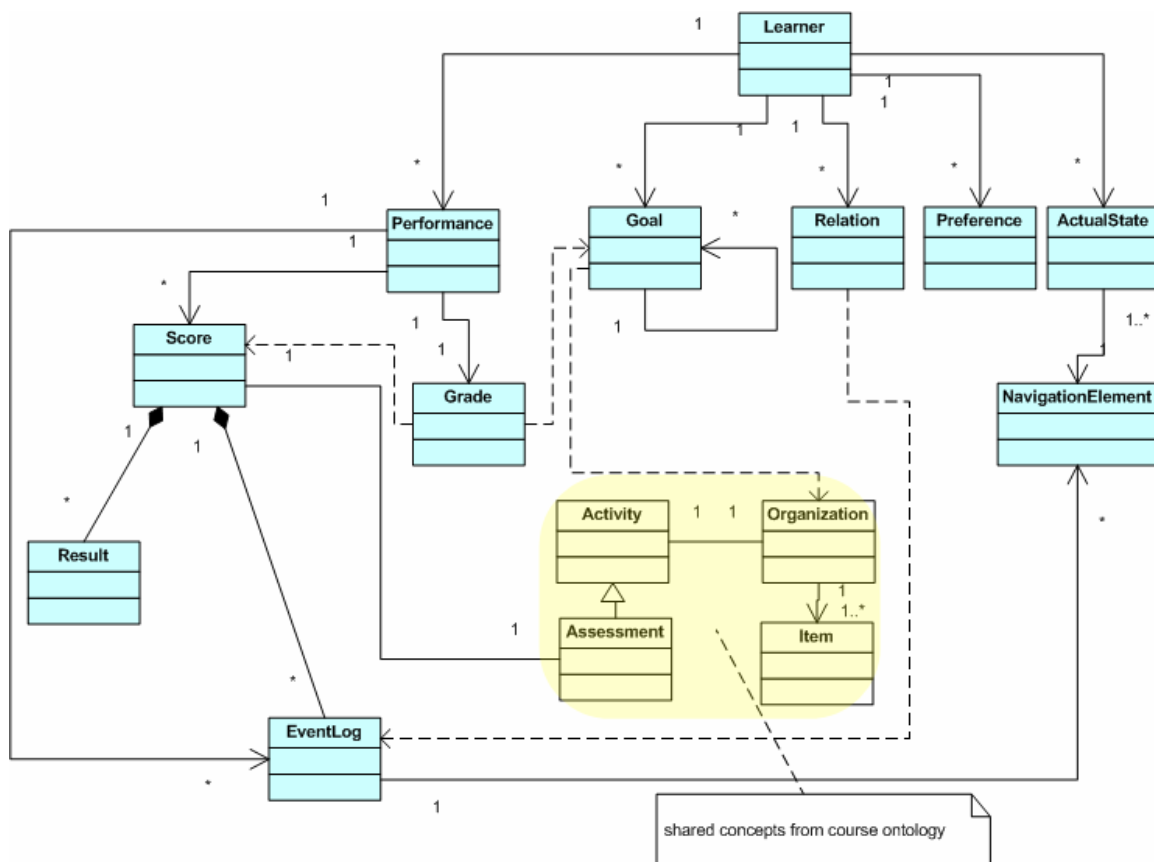
Studentski model je osnova za personalizaciju procesa učenja. Kompleksnost studentskog modela je proporcionalna broju funkcija koju sistem pruža studentu i potrebama za adaptaciju sadržaja učenja. Pored rezultata testova, sistem prikuplja veliku količinu podataka tokom sesija studenata sa sistemom. Ovi podaci mogu da budu vrlo neprecizni, dvosmisleni, i čak kontradiktorni. Da bi model aproksimirao realno stanje studenata u posmatranom kontekstu učenja, često je neophodno da se obavi rezonovanje nad svim tim podacima. Zbog ugnježenosti studentske aplikacije, studentski model je podeljen na dva dela.

Deo koji je predefinisani i nalazi se u IS domaćina. Ovaj model se koristiti samo delimično, isključivo radi provere identiteta studenta, pošto čuva osnovne podatke studenta. Na primer, identifikacioni podaci, afilatorni podaci, kompetencije i mnogi drugi (detalji su opisani u poglavlju 3.1). Po tom kriterijumu, ovaj deo je sličan bilo kom standardnom studentskom modelu (Poglavlja 3.1.1. – 3.1.4.).

Drugi deo studentskog modela (Ilustracija 109) je dizajniran za potrebe procesiranja učenja u nastavničkoj i studentskoj aplikaciji, sa ciljem da se uključe svi podaci, implicitni i eksplicitni, a koji su povezani za konkretnog studenta.

Pri upisu studenta (kroz sistem domaćina) on postaje član odgovarajuće nastavne grupe. Na osnovu pripadanja grupi, sistem određuje predmete (kurseve) koje student pohađa. Predmeti su opisani kurikulumima i silabusima. Za svaki predmet definišu se ciljevi. Postoje ciljevi (*Goals*) zahtevani od sistema (zahtevani minimumi), i ciljevi koje student eksplicitno unosi. Na osnovu unosa studenta, sistem dobija podatak o njegovim motivima, koje kasnije koristi za njegovo vođenje i pružanje pomoći u učenju. Ciljevi mogu da se dekomponuju u podciljeve koji se vezuju na strukturu predmeta (činioce).

Napredovanje studenta (*Grade*) se razmatra u kontekstu ciljeva. Ono se dobija iz rezultata studenata i važno je za donošenje globalnih odluka ostudentu. Na primer, student je završio n -tu godinu studija kada u skupu završenih obaveza (O_n) prikupi sve uslove predviđene kurikulumom. To znači da je uslov za upis $(n+1)$ godine podskup skupu završenih obaveza ($O_{np} \subset O_n$). Izvedeni podatak o napredovanju omogućava sistemu da izbegne procesiranje velike količine *sirovih* podataka svaki put kada postoji korisnički ili sistemski zahtev za podatkom *Grade* studenta.

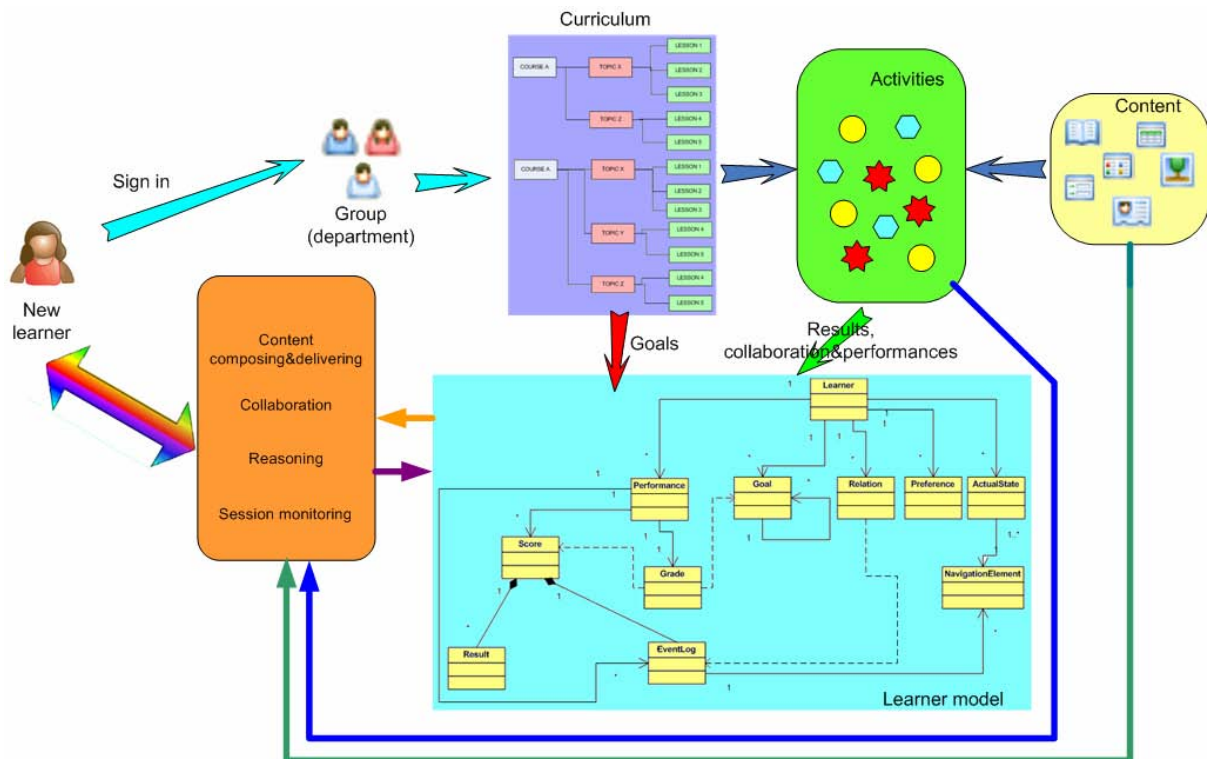


Ilustracija 108: Strudentski model

Pri interakciji između studenta i sistema (u kontekstu specificiranog predmeta), sistem meri razliku između ostvarenih ciljeva (izraženih kroz aktuelne performanse korisnika i napredovanje) i planiranih ciljeva (instance klase *Goal*). Razlika se dalje koristi za

personalizaciju sadržaja učenja, i za pomoć studentu u procesu učenja [76]. Klasa *Relation* enkapsulira kolaboracione podatke – zaključke rezonovanja iz studentskih logova. Student može da promeni svoje preference (npr. interesovanja, izgled interfejsa), svaki put kada koristi aplikaciju. Klasa *Performance* daje konačan profil studenta na osnovu podataka iz povezanih klasa *Grade* i *Score*. Ova klasa je namenjena za izvoz podataka van sistema (npr.ka drugim sistemima, ili za izdavanje suplement diplome, i sl.). Klasa *Score* je osnovna mera performanse studenta.

Kao što je prethodno objašnjeno (Poglavlje 3.1), studentski model predstavlja osnovnu komponentu koja omogućava da sistem adaptira sadržaj i personalizuje pomoć koju pruža studentu. Proces adaptacije je dinamičan, i izvodi se dok god traje interakcija korisnika (studenta) sa sistemom. Funkcioniše po principu povratne sprege (Ilustracija 110). Pošto student pripada nastavnoj grupi, koja radi po predefinisanim kurikulumu, ciljevi koji se dodaju u model su definisani u kurikulumu. Nastavnik operacionalizuje sadržaj kurikuluma kroz aktivnosti koje student treba da realizuje u toku učenja. Kroz aktivnosti, student koristi sadržaje učenja, a sistem meri njegovo napredovanje. Ovim podacima (rezultati, performanse, saradnja sa drugim korisnicima i sl.) se ažurira studentski model.



Ilustracija 109: Interakcija studentskog modela sa ostalim entitetima u sistemu

Delovi sistema koji su namenjeni za komponovanje, prezentaciju sadržaja, za saradnju između korisnika i za praćenje sesije, koriste podatke iz studentskog modela da bi adaptirali ponašanje sistema i sadržaje učenja prema individualnom studentu. Student dalje preduzima akcije koje generišu nove podatke, kojima sistem iznova ažurira model i ciklus adaptacije se ponavlja.

6.1.2.6 Paket *DataBrokers*

Ovaj paket predstavlja skup klasa namenjenih serijalizaciji – deserijalizaciji podataka studentske aplikacije. Konceptualno je sličan istoimenom paketu nastavničke aplikacije. Razlika kod studentske aplikacije je u dodatnim klasama – brokerima, namenjenim za posredovanje u razmeni podataka studentskog modela i baze podataka. Pored navedenih brokera, kreirani su i brokeri posebne namene. Jedan od njih je namenjen za pamćenje studentskih logova (Ilustracija 111).

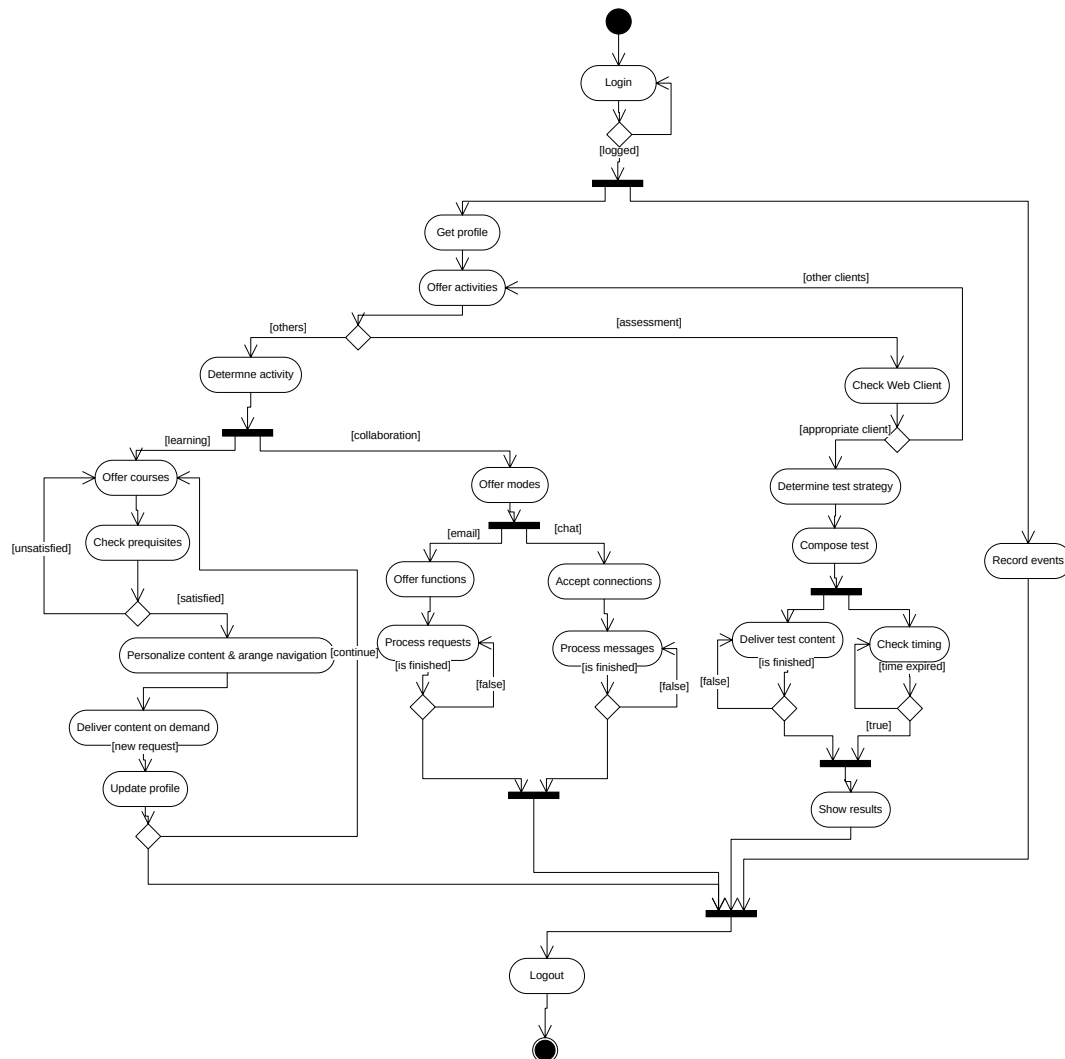
ID	Student_ID	Item_ID	Resource_ID	Module	StartTime	EndTime	Status
1	4770	7	3	PDF	1222245450231	1222245481893	1
2	4770	6	2	PDF	1222245481958	1222245492786	1

Ilustracija 110: Tabela logova

Pored identifikatora zapisa i studenta u logu se pamti nastavni činilac (kontekst u kome se akcija odigrala), resurs koji je bio korišćen (npr. materijal za učenje, samotestiranje, ili testiranje), modul kroz koji se resurs koristio (u primeru na slici korišćen je modul za prikazivanje PDF fajlova), vremena početka i kraja aktivnosti i status aktivnosti (da li je akcija završena regularno, ili ne).

6.1.3 Dijagram aktivnosti studentske aplikacije

Na osnovu slučajeva korišćenja studentske aplikacije (Poglavlje 6.1.1), postoje tri osnovne grupe aktivnosti kroz koje student koristi sistem: učenje, kolaboracija i proveravanje. Ove tri globalne aktivnosti mogu da se prikažu jedinstvenim dijagramom aktivnosti (Ilustracija 112). Sledi opis navedenih aktivnosti, korišćenjem dijagrama aktivnosti.



Ilustracija 111: Osnovne studentke aktivnosti u eksploataciji sistema

Nakon logovanja studenta, sistem simultano obavlja dve aktivnosti: formira studentski model (aktivnost *Get profile*) i registruje događaj da se korisnik ulogovao (aktivnost *Record Events*). Sve vreme dok traje sesija između studenta i sistema, beleže se akcije studenta (snimaju se događaji). To znači da aktivnost imenovana kao *Record Events* traje sve dok traje studentska sesija.

Sledeća obavezna aktivnost studentske aplikacije je da ponudi studentu funkcionalnosti (*Offer Activities*). Na osnovu studentskog izbora, postoje dva alternativna toka događanja. Za slučaj da student bira proveru, potrebna je dodatna provera klijentske aplikacije. U slučaju da provera nije prošla, sistem ne dozvoljava proveru i vraća studenta na ponovno biranje funkcionalnosti.

Drugi tok događanja je za sve ostale aktivnosti izuzev proveravanja. U ovom toku postoje dve grane za dve funkcionalnosti: učenje i kolaboracija. Aktivnosti obuhvaćene u ove dve grane mogu da se izvode simultano. To znači da ako student, na primer, uči (koristi) nastavni sadržaj, on istovremeno može da se konsultuje sa drugima u vezi istog.

6.1.3.1 Aktivnosti učenja

Aktivnosti učenja započinju izborom predmeta, zatim nekog činioca (*Item*) predmeta za učenje (prikazano kao jedna aktivnost *Offer Courses*). Sistem zatim ispituje da li student zadovoljava sve predušlove potrebne za pristup resursima odabranog činioca učenja (detaljnije u poglavlju 6.1.5). U slučaju da kriterijumi nisu ispunjeni, aplikacija informiše studenta o njegovim performansama i predušlovima koje treba još da ispuni kako bi pristupio resursima željenog činioca. Student nakon toga samo može da ponovo bira predmete i njihove činioce (povratni tok na uslov *Unsatisfied*).

Ako su svi predušlovi ispunjeni, sistem koristi podatke iz modela studenta i vrši personalizaciju sadržaja i navigacije (aktivnost *Personalize Content & Arange Navigation*). Nakon komponovanja, aplikacija isporučuje zahtevane sadržaje studentu (aktivnost *Deliver Content on Demind*). Ako je student obavio samotestiranje (u slučaju da su definisana pitanja za samotestiranje iz odabranog činioca), sistem ažurira podatke u studentskom modelu (aktivnost *Update Profile*). Time je završen jedan ciklus aktivnosti učenja. Ako student želi da nastavi, ponovo se usmerava na izbor predmeta i činioca. Čitav proces se ponavlja sve dok student to zahteva.

Istovremeno sa učenjem, studentska aplikacija omogućava korišćenje kolaborativnih alata. Kolaboracija studenata nije moguća samo u fazi samotestiranja. U svim ostalim fazama kolaborativne funkcionalnosti su u potpunosti dostupne. Ovaj detalj se rešava na klijentskoj strani aplikacije, korišćenjem tzv. modalnog interfejsa, koji ne dozvoljava pristup ostalim kontrolama dok je aktivan.

6.1.3.2 Kolaborativne aktivnosti

Sistem nudi studentu različite kolaborativne alate (aktivnost *Offer Modes*). Na dijagramu su predstavljena dva moda: e-mail i chat, kako bi se prezentovale aktivnosti asinhrona i sinhrona kolaboracije. Termin komunikacija s namerom nije korišćen u materijalu, pošto je komunikacija u sistemu eksplicitno kontekstualna. To znači da aplikacija ima interni mail i chat servis koji pruža studentima mogućnost da, na primer, u slučaju izučavanja nekog predmeta (kursa), stupi u sinhronu komunikaciju (chat) sa ostalim korisnicima (studentima, nastavnicima) koji istovremeno koriste resurse istog predmeta. S druge strane, mail je takođe kontekstualan, jer je omogućena komunikacija samo sa studentima koji se nalaze u istoj nastavnoj grupi (npr. fakultet, smer, godina studija) i nastavnicima koji predaju dostupne predmete. Komunikacija je, dakle, u sistemu korišćena kao sredstvo za ostvarivanje saradnje među svim akterima procesa učenja.

Pored navedenih kolaborativnih alata, postoje još forumi i dodavanje komentara. Forumi su tematski, prema sadržajima koji se izučavaju. Teme definišu nastavnici – eksplicitno, ili implicitno.

- Eksplicitno – kada se pred nastavnu grupu postavi neki konkretan zadatak, ili problem, koji treba da se reši.
- Implicitno – postavljanje teme obavlja sistem, automatski, na osnovu definisanih aktivnosti od strane nastavnika.

Posredstvom foruma, nastavnici se obraćaju nastavnoj grupi, a svi zahtevi od studenata prema nastavnicima postaju transparentni.

Studenti dodaju komentare kao beleške, ili kao vlastitu anotaciju sadržaja (poglavljje 6.1.1.1, SK *CommentManipulation*). Ako su komentari objavljeni (podrazumevana vrednost je da su privatni, tako da student mora eksplicitno da ih „objavi“), oni postaju sredstvo za kolaboraciju. Preko komentara, student pomaže drugima da lakše shvate sadržaj na koji se komentar odnosi. S druge strane nastavnik može da stavi dodatne komentare na svoj resurs učenja, kako bi preciznije objasnio, ili nadopunio sadržaj, i time pomogao studentima.

6.1.3.3 Aktivnost proveravanja

U slučaju proveravanja, pre nego što se test isporuči studentu, aplikacija proverava pedagogiju koja je primenjena na odabranu proveru (aktivnost *Determine Test Strategy*), i na osnovu toga komponuje test (aktivnost *Compose Test*). Pored osnovnih podataka provere (poglavljje 6.1.2.2), sistem generiše test odgovarajuće težine (npr. da li se radi o proveru osnovnog, ili naprednog nivoa znanja).

Nakon komponovanja, sistem isporučuje test zadatke (pitanja) studentu (aktivnost *Deliver Test*) i simultano proverava i evidentira vreme rada (aktivnost *Check Timing*) (registruje se vreme predaje svakog odgovora, ili rešenja). Ove dve aktivnosti su iterativne, i traju sve dok vreme testiranja ne istekne, ili student ne reši sve postavljene zadatke. Nakon toga sistem prikazuje rezultate u formi koju je definisao nastavnik (aktivnost *Show Results*). U zavisnosti od definisane pedagogije, studentu mogu da se prikažu samo zbirni rezultati, zbirni i detaljni, sa obrazloženjem za svako pitanje, ili samo statistika proveravanja (vreme, procentualna tačnost i netačnost), a pravi rezultati tek nakon verifikacije od strane nastavnika.

Eksploatacija sistema (sesija) se završava kada se student izloguje iz aplikacije (aktivnost *Logout*). Regularno, potrebno je da se student eksplicitno izloguje pomoću forme, ili zatvori prozor klijentske aplikacije, čime se u sistemu generiše događaj kraja sesije. U slučaju da student zaboravi da učini jednu od dve pomenute akcije, sistem se štiti od zaboravljenih sesija mehanizmom maksimalnog vremena neaktivne sesije. Klijentska aplikacija meri vreme između akcija korisnika (pomeranje, kliktanje mišem, korišćenje tastature). Ako postoji pasivnost studenta, veća od maksimalnog zadatog vremena, sesija se nasilno prekida i oslobađaju se sistemski resursi.

6.2 Adaptivna navigacija i sadržaji

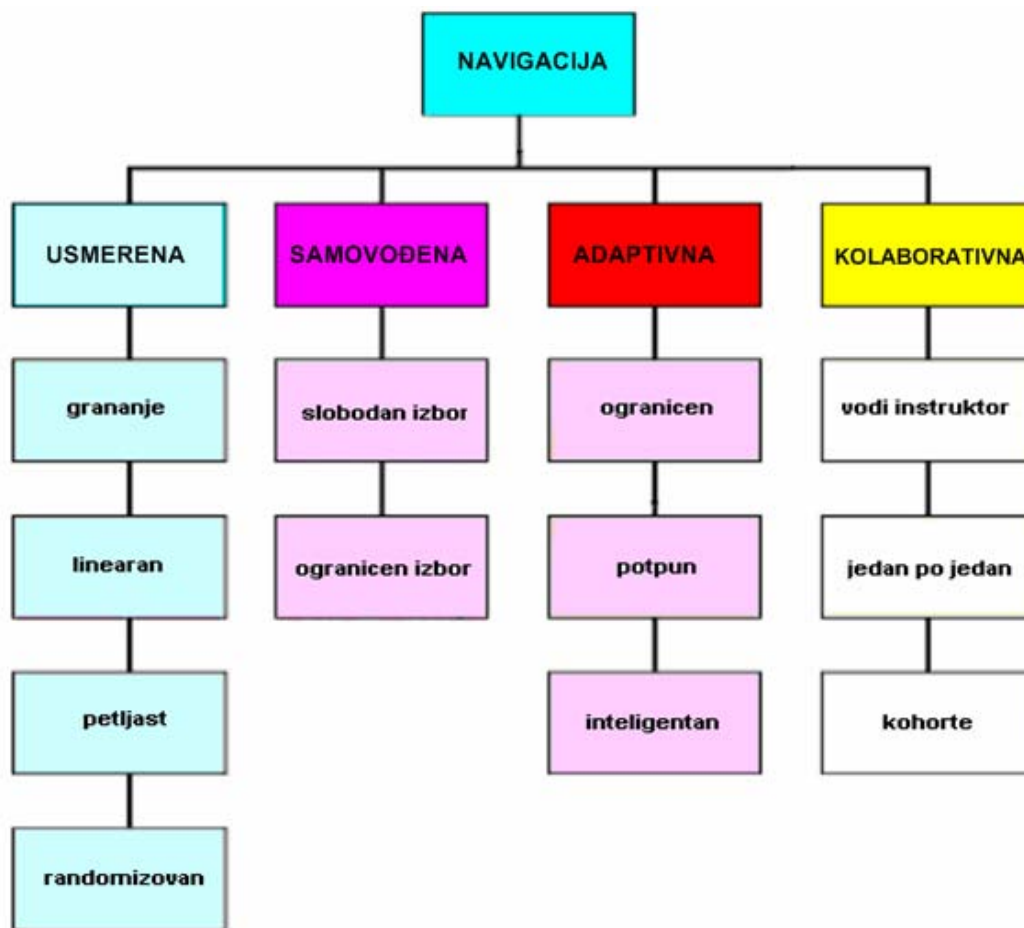
Postoje različiti tipovi navigacije po sadržajima učenja [77] (Ilustracija 113). Osnovne grupe su: usmerena navigacija, samovođenje, adaptivna navigacija i kolaborativna navigacija. Kod usmerene navigacije sistem u potpunosti kontroliše navigacione putanje korisnika.

Usmereno vođenje je unapred programirano – na isti način se odvija za sve korisnike. Samovođenje ima dva režima: korisnik ima potpunu slobodu u pristupu sadržajima; u drugom režimu, sistem stvara privid kod korisnika da je navigacija potpuno slobodna ograničavanjem izbora (sklanjanjem ili sakrivanjem linkova). Kao kod usmerene navigacije, vođenje je unapred programirano.

Kolaborativna navigacija se ostvaruje kroz kolaborativne aspekte učenja. Najčešći slučaj je da instruktor sadržaje čini dostupnima kroz definisanje aktivnosti. Određivanjem

vremenskog intervala aktivnosti, sadržaji koji su uključeni u aktivnosti bivaju vidljivi u predefinisanom vremenu. Aktivnosti mogu da grupišu sadržaje u tzv. *kohorte* – tako da su sadržaji iste kohorte vidljivi studentima. Alternativni način je da se definišu intervali (aktivnosti) sa delimičnim preklapanjima za svaki pojedinačni sadržaj, tako da se time postiže sekvencijalnost pristupa.

Adaptivna navigacija je, drugim rečima, personalizovana navigacija. Može da bude ograničeno adaptivna, potpuno adaptivna i inteligentna (detalji u poglavlju 3.4.3.2). Adaptivna navigacija se dinamički podešava prema konkretnom studentu na osnovu njegovog aktuelnog stanja, koje se čuva u studentskom modelu. Ona je potpuna ako su implementirani svi pod-tipovi (poglavlje 3.1), odnosno delimična ako su implementirani samo neki tipovi navigacije. Intelligentna navigacija podrazumeva inteligentno vođenje studenta od strane sistema, kroz prostor znanja (kao što je to slučaj u ITS-ovima – poglavlje 3.4.1)



Ilustracija 112: Načini definisanja redosleda resursa učenja

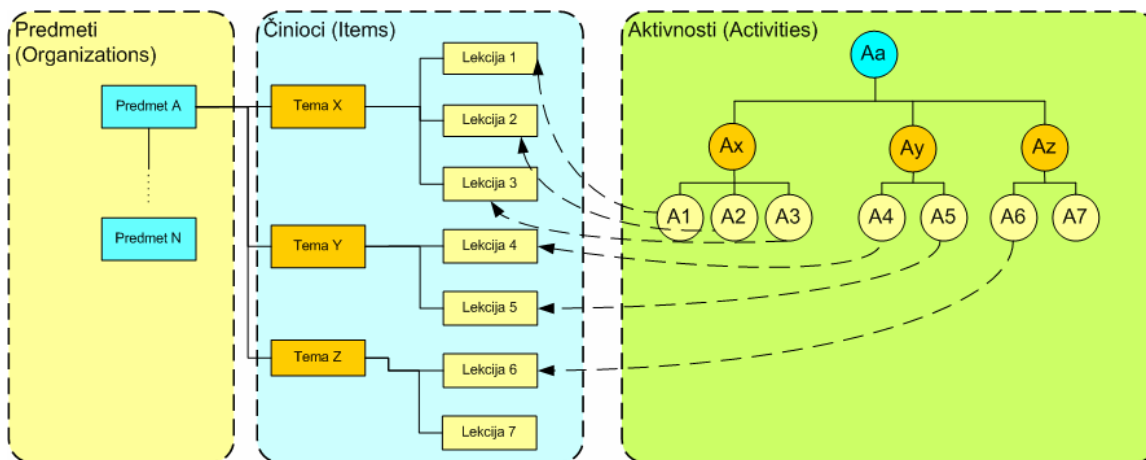
IMS SS specifikacija (poglavlje 3.2.3.3), korišćena kao osnov za rešavanje navigacije kod većine LMS sistema (bilo da se radi o samostalnoj implementaciji, ili u okviru SCORM standarda – poglavlje 3.2.4), obuhvata u potpunosti dva tipa – usmerenu navigaciju i samovođenje.

Standardnom nije predviđena mogućnost kolaborativne navigacije. Moguća je delimična (ograničena) adaptivna navigacija, na bazi pravila za formiranje redosleda materijala. Ova pravila se zasnivaju se na ispitivanju rezultata studenata, a konsekvencu svakog pravila je da student ispunjava/ne ispunjava preduslove za pristup željenom sadržaju.

6.2.1 Parametri aktivnosti

Aktivnost je koncept koji se uvodi radi rešavanja navigacione adaptivnosti (pominjan u poglavlju 6.1.2.4). Predmet (*Organization*) i činilac predmeta (*Item*) su koncepti koji su uvedeni radi struktuiranja znanja i organizovanja procesa učenja. Učenje može da se posmatra kao višedimenzioni proces. Ono sadrži vremensku dimenziju (učenje ima početak, kraj i trajanje, rokove za završavanje zadatih obaveza), saznavnu dimenziju (učenjem student stiče novo znanje čiji kvalitet može da se predstavi skalarom, kao razlika prethodnog i novog nivoa znanja), socijalnu dimenziju (studenti su grupisani, upućeni učenje jedni na druge, na nastavnike, saradnike i eksperte) i prostornu dimenziju (proces učenja se realizuje u fizičkom, i/ili virtuelnom prostoru).

Posredstvom činilaca, studenti pristupaju sadržajima učenja. Struktuiranjem činilaca, struktuirani su i sadržaji. To znači da se navigacija odnosi na kretanje po strukturi predmeta. Činioci, međutim, ne mogu da sadrže podatke potrebne za rešavanje adaptivne navigacije. Da bi se navigacija personalizovala, koriste se *Aktivnosti* (Ilustracija 114), koje takođe oslikavaju strukturu predmeta. Aktivnosti se definišu za kompletnu strukturu predmeta (na primer, aktivnost A_x definisana je za temu x).

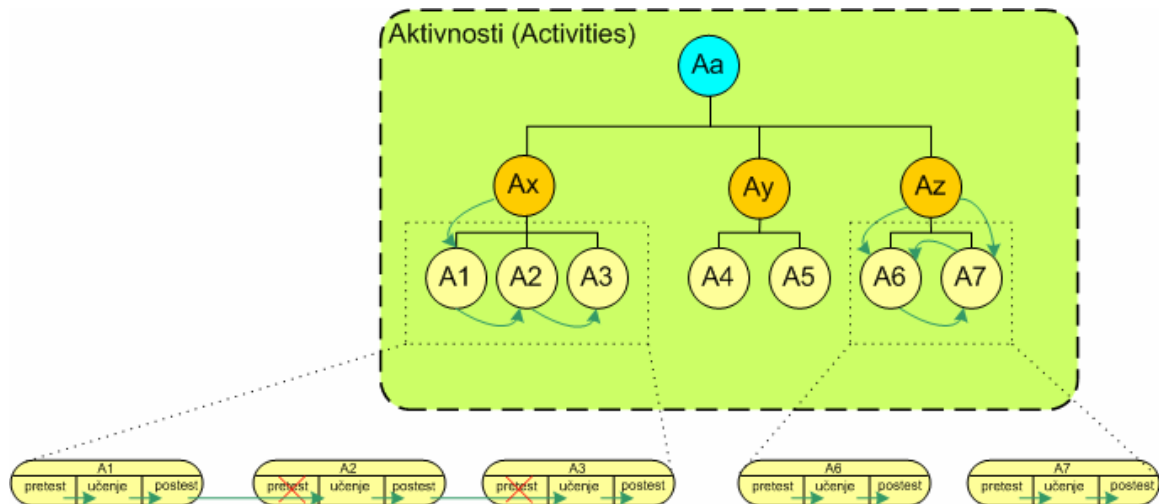


Ilustracija 113: Odnosi predmeta, činilaca predmeta i aktivnosti

Studenti više ne pristupaju direktno činionicima predmeta, već posredstvom aktivnosti. Ova indirekcija omogućava da se na nivou aktivnosti definišu specifični podaci vezani za način korišćenja nastavnih sadržaja. Elementarni skup aktivnosti koji treba da se definiše za svaki predmetni činilac obuhvata:

- Prethodni test (u daljem tekstu pretest) – namenjen merenju predznanja studenta
- Učenje – proces korišćenja resursa konkretnog činioca
- Završni test (u daljem tekstu posttest) - namenjen merenju znanja studenta, stečenog u prethodnoj aktivnosti (učenju)

Ove tri aktivnosti, iako su osnovne, nisu obavezne (Ilustracija 115). Na primer, ako je navigacija sekvencijalna, potpuno ograničena, umesto rezultata pretesta, sistem može da koristi rezultate postesta prethodnog činioca kao meru ulaznog znanja studenta za sledeći činioc (na primer aktivnost A2 koristi završni test aktivnosti A1).



Ilustracija 114: Elementarne aktivnosti kod različitih tipova navigacije

Ako je navigacija slobodna, ili bez ograničavanja vidljivosti linkova, (primer aktivnosti Az, A6 i A7), pretest je neophodan, jer student svakoj aktivnosti može da pristupi direktno. Pored toga, pretest je značajan za ocenu predznanja studenta pre početka izučavanja predmeta. Samo na taj način mogu da se dobiju validni podaci o efektima upotrebe sistema.

Korišćenjem mehanizama pretesta i završnog testa, sistem je u stanju da meri relativni nivo znanja studenta. Time se prevazilazi osnovni nedostatak prekrivajućeg modela (poglavlje 2.2.2.1), u kome se meri samo apsolutni nivo znanja studenta. Ukupno znanje se izražava kao agregacija (skup) relativnog i apsolutnog izmerenog znanja (izraz 20).

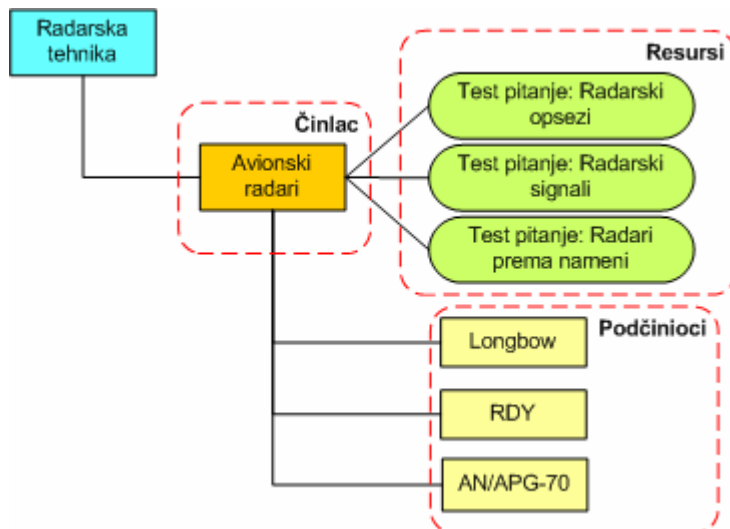
$$R_{akt} = \{R_{abs}, R_{rel}\}, \quad (20)$$

gde je apsolutni mereni nivo znanja studenta $R_{abs} = R_{max} - R'_p$, a relativni $R_{rel} = R'_p - R_p$, pri čemu je R_p rezultat pretesta, a R'_p rezultat postesta. Na ovaj način sistem (preciznije – studentski model) poseduje kompletnu sliku o znanju, o napredovanju, angažovanju i motivaciji svakog studenta.

Nivoi znanja se kvantitativno izražavaju poenima (bodovima). Svaki student kroz savladavanje sadržaja predmeta (kursa) sakuplja poene. Svaki predmetni činilac za koji su definisana pitanja za samotestiranje, sadrži izvedeni podatak o prolaznom i maksimalnom nivou znanja. Svako pitanje za samotestiranje nosi određeni broj bodova. Maksimalan broj poena koje student može da sakupi za jedan činilac predmeta predstavlja sumu bodova svih pitanja za samotestiranje, definisanih za posmatrani činilac.

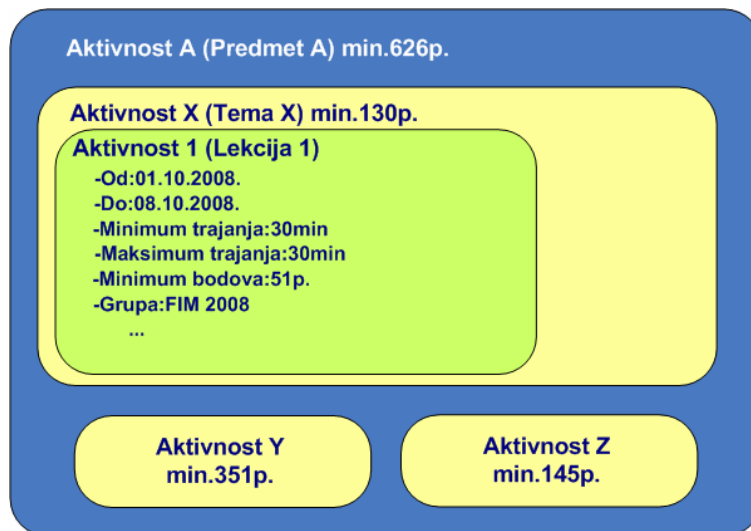
Za razliku od činilaca (*Item-a*) SCORM i IMS standarda koji, ako su struktuirani, ne mogu da sadrže resurse za učenje i proveravanje, u predloženom rešenju svi činioici imaju identično ponašanje. To znači da struktuirani činioici, pored referenci na podčinioce, i sami mogu da sadrže resurse. Ova osobina pruža mogućnost nastavnicima da uvodni, ili

pripremni sadžaj pridruže direktno činiocu (koji je struktuiran). Na primer (Ilustracija 116), pre prelaska na novu tematsku celinu (Avionski radari), nastavnik želi da ispita predznanja studenata iz neke oblasti koja se izučavala van posmatranog činioca, ili predmeta (test pitanja iz osnovna radarstva).



Ilustracija 115: Primer struktuiranog činioca (Avionski radari) sa direktno vezanim test setom

Kod struktuiranih činilaca rezultat studenta predstavlja kumulativni zbir bodova podčinilaca (sume bodova podčinilaca se udružuju) i bodova iz testova direktno vezanih za struktuirani činilac (Ilustracija 117). To znači da sistem automatski određuje maksimalan broj bodova po činiocu. Definisanje zadovoljavajućeg nivoa znanja – broja bodova, predstavlja deo nastavne pedagogije. U slučaju da nastavnik eksplicitno ne odredi minimum, sistem koristi predefinisani podatak da je minimum zadovoljenja 51% maksimalnog broja bodova. Kriterijumi zadovoljenja na višim nivoima, izvode se na osnovu kriterijuma definisanih na nivou elementarnih činilaca.



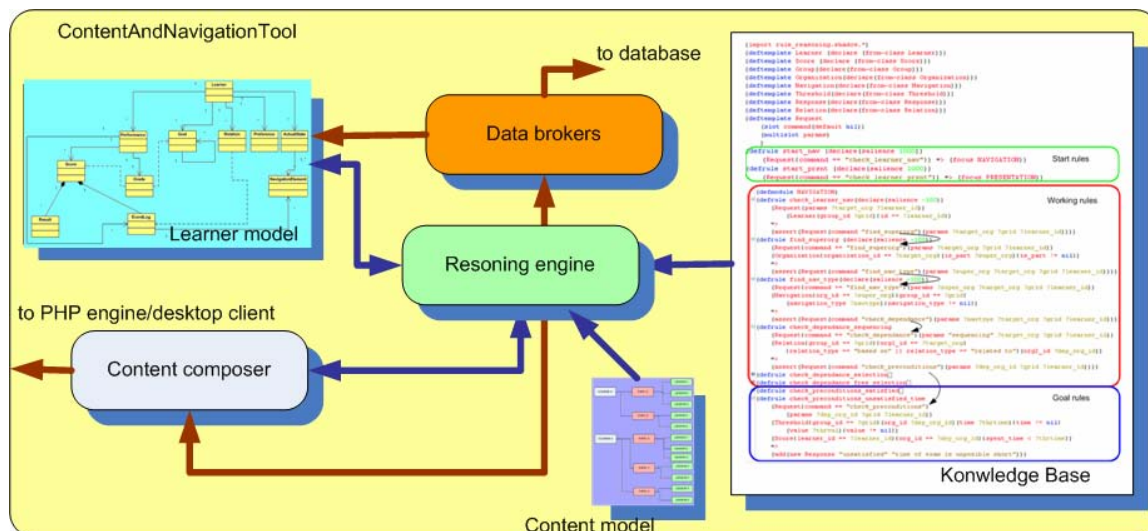
Ilustracija 116: Bodovanje činilaca predmeta

Povezivanjem kriterijuma provere znanja za aktivnosti, omogućeno je da se za različite nastavne grupe definišu različiti kriterijumi za isti nastavni činilac. Ista karakteristika je

primenjiva i za ostale podatke vezane za aktivnost (poglavlje 6.1.2.2). Vremenski podaci aktivnosti (početak, kraj, najkraće i najduže trajanje) takođe utiču na navigacionu adaptivnost sistema. Ovi podaci omogućavaju da se navigacione putanje optimizuju, ali isto tako se koriste u dijagnostici studentskog nepoznavanja. Na primer, nezadovoljavajući rezultati studenta na samotestiranju trigeruju proveru studentskog log zapisa radi utvrđivanja vremena i navigacionih putanja. Ovi podaci se propuštaju kroz sistem pedagoških pravila, (ako su definisana), a zaključak se koristi kao dijagnoza za eventualnu pomoć studentu (u vidu preporuka, ograničenja navigacije, vraćanje na činioce kod kojih je ustanovljeno neznanje, i sl.).

6.2.2 Dizajn u cilju adaptivnosti

Adaptivnost studentske aplikacije je implementirana u alatu za učenje (*ContentAndNavigationTool*, poglavlje 6.1.2.1) i zasniva se na zaključivanju o svakom pojedinačnom studentu. Ovaj proces je kompleksan. U zaključivanju se uključuju istorijski podaci studenta (Ilustracija 118), koji se dobijaju posredstvom brokera iz BP (*Data Brokers* – poglavlje 6.1.2.6). To su podaci o svim aktivnostima studenta vezanim za posmatrani predmet, ili predmetni činioc (poglavlje 6.1.3). Na primer, obuhvaćeni su podaci učenja, provera i samotestiranja (rezultati, vremena trajanja, greške, navigacione putanje i sl.). Pored navedenih podataka, u zaključivanje su uključeni aktuelni podaci, koji se trenutno nalaze u studentskom modelu (poglavlje 6.1.2.5). Pošto se rezonovanje odnosi na komponovanje navigacije i sadržaja, podaci o sadržaju su takođe uključeni (*Content model*). Ovi podaci predstavljaju ulazne vrednosti u mašinu za zaključivanje (*Reasoning Engine*). Rezonovanje se vrši na osnovu entiteta (pravila i šablona) baze znanja (*Knowledge Base*). Aplikacija učitava bazu znanja u radnu memoriju mašine za zaključivanje pre prosleđivanja ulaznih podataka.



Ilustracija 117: Arhitektura studentske aplikacije

Tokom procesa učenja, sistem domaćina prosleđuje studentski zahtev ovom alatu. *Content composer* prihvata prosleđeni zahtev i traži od mašine za rezonovanje da donese odluku o zahtevanom sadržaju. Mašina za rezonovanje razmatra aktuelno stanje studentskog modela, zajedno sa modelom sadržaja i prosleđenim zahtevom. Zaključivanje se obavlja na osnovu pravila baze znanja. Kao proizvod rezonovanja

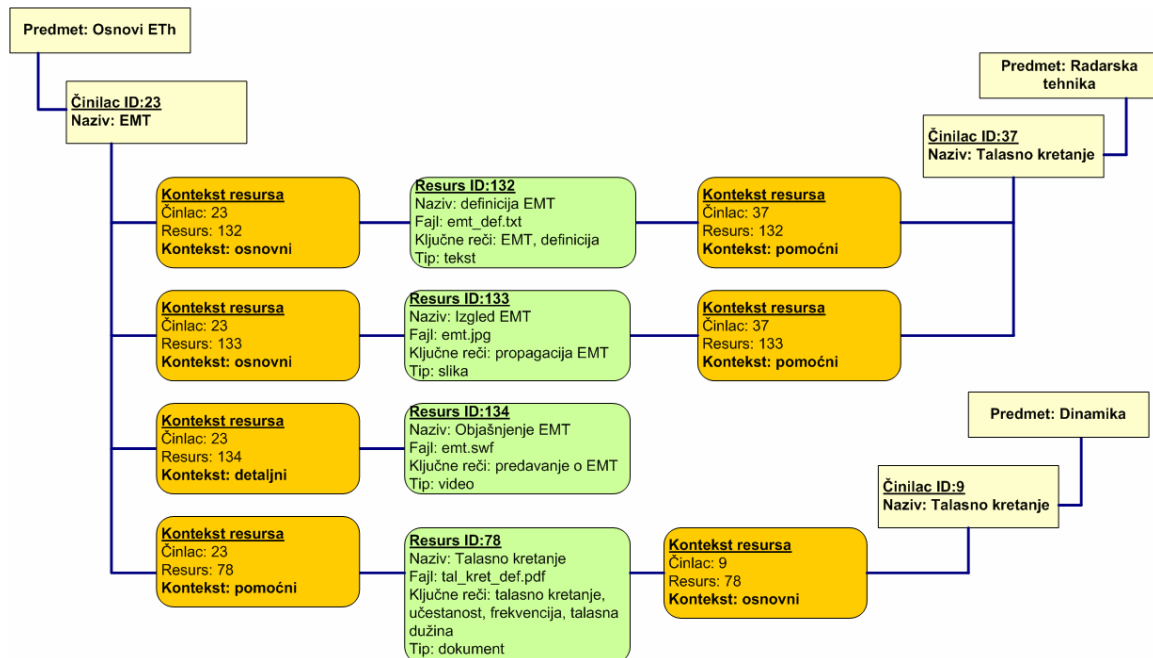
dobijaju se izlazni podaci (struktuirani kao objekti), koje koristi komponenta *Content composer* i dinamički komponuje zahtevan sadržaj i navigaciju.

6.2.3 Proširivanje standardne strukture sadržaja učenja

Da bi se omogućila adaptacija sadržaja učenja, predmeta treba da ima vrlo razgranatu strukturu, a resursi (fajlovi različitog tipa, npr. pdf dokumenta, tekstovi, ilustracije, video i audio zapisi) treba da budu fragmentirani, opisani i semantički povezani. Na taj način, fragmentima se može manipulirati u smislu dodavanja, skrivanja, izmene, sortiranja (poglavljje 3.4.3.1) u dinamički kreiranim nastavnim sadržajima koji se dostavljaju korisniku (studentu). Svaki predmetni činilac može da ima pored pod-činilaca i resurse (sadržaje učenja). Dinamičko komponovanje sadržaja nije moguće ako se resursi ne klasifikuju prema nameni u konkretnom činiocu. Aktuelni standardi za komponovanje sadržaja nemaju eksplicitnu specifikaciju gde i kako treba da se implementira mehanizam adaptacije sadržaja. Razlog tome je što u adaptivnim sistemima postoje specifičnosti implementacije, tako da standardi za njih nisu razvijani, a kod sistema za upravljanje učenjem (za koje su internacionalni standardi razvijani), postoji samo delimična navigaciona adaptacija.

Iz navedenog razloga, u model nastavnog sadržaja dodat je još jedan koncept – *kontekst resursa* (Ilustracija 119), koji sadrži identifikator činioca, identifikator resursa, i podatak o načinu korišćenja resursa (kontekst). U predloženom rešenju definisana su tri konteksta:

- Osnovni kontekst – resurs se prikazuje uvek kada se potražuje činilac
- Detaljni kontekst – resurs se prikazuje kada činiocu pristupa napredni student
- Pomoćni kontekst - resurs se prikazuje kada činiocu pristupa student koji ima manjkavosti u znanju iz izabranog činioca



Ilustracija 118: Različiti tipovi resursa

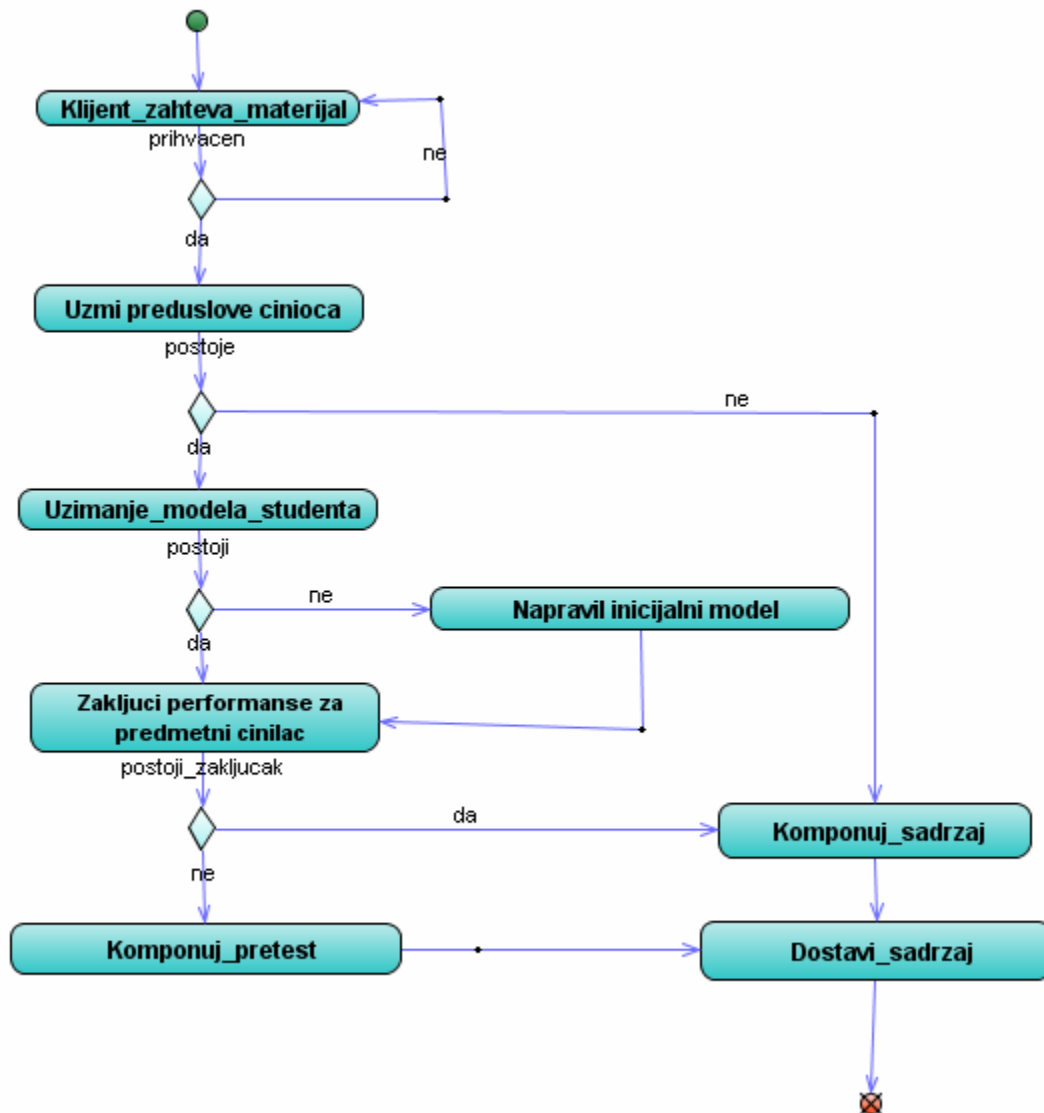
To znači da je ovakvim rešavanjem selekcije nastavnih materijala učenje klasifikovano u dve kategorije: osnovnu i naprednu. Pomoćni kontekst treba da omogući studentima da mogu da savladaju osnovni nivo izučavanja korišćenjem mehanizama pomoći sistema.

Predstavljeno strukturno proširenje je osnova za adaptivno komponovanje sadržaja u predloženom rešenju. Adaptacija se odvija u studentskoj aplikaciji kroz posebne aktivnosti sistema, svaki put kada student zahteva nastavni sadržaj posredstvom klijentske aplikacije.

6.2.3.1 Dijagram aktivnosti

Sistem dinamički komponuje sadržaje u nekoliko faza (aktivnosti) (Ilustracija 120). Materijal se komponuje po zahtevu klijenta. Klijentska aplikacija (poglavljje 4.2.1) omogućava studentu, posredstvom mehanizma navigacije i strukture predmeta, da pristupi resursima učenja (materijalima) koji su povezani za predmetni činilac (npr. temi, ili lekciji, poglavlja 3.2.3.2 i 5.3.2.2). Nakon prihvatanja zahteva za materijal od strane klijenta, serverska aplikacija prikuplja podatke o preduslovima pristupanja činiocu. Ako preduslovi nisu specificirani, serverska aplikacija komponuje sadržaj na podrazumevani način (jednostavno raspoređivanje svih materijala po unapred utvrđenom šablonu).

Ako postoje preduslovi, sledeći korak u serverskoj aplikaciji je da se model (podaci) studenta koji je poslao zahtev, dopuni informacijama (rezultatima testiranja i samotestiranja) potrebnim za odlučivanje o komponovanju. Ako ove informacije ne postoje (npr. radi se o novom studentu), aplikacija kreira inicijalni model, koji se popunjava podacima šireg konteksta od predmetnog.



Ilustracija 119: Dijagram aktivnosti - adaptivno komponovanje sadržaja

U svakom slučaju sistem zaključuje o zahtevanom sadržaju. Ako zaključak ne postoji (preciznije, ne postoji dovoljan nivo prihvatljivosti zaključka), umesto sadržaja komponuje se pretest, koji treba da omogući profilisanje studenta.

6.2.4 Šabloni i činjenice – senke

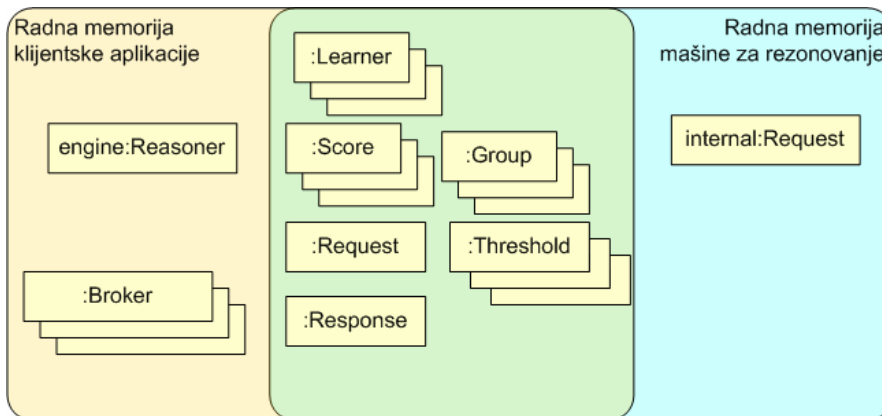
Bazu znanja za potrebe adaptacije čine pravila i šabloni (*templates*). Šabloni predstavljaju strukture podataka slične objektima, koji podržavaju sve osnovne tipove podataka, a imaju ugrađene specijalne metode – demonske procedure. Namena šablona je da prihvataju ulazne podatke (fakta, ili činjenice) za rezonovanje, da grade premise i konsekvence pravila i da prihvataju rezultate rezonovanja. Pored toga, pogodni su za čuvanje privremenih podataka i međurezultata zaključivanja. Koncept *template*-a je preuzet iz ekspertskih sistema koji rade na principu okvira (*frames*) [136]. Osnovni

koncepti predikatske logike (univerzalni i egzistencijalni kvantifikatori) su u ES implementirani preko šablona.

```
(deftemplate Request
  (slot command (default nil) (type STRING))
  (multislot params)
)
```

Fragment 20: Definisanje šablona u CLIPS skript jeziku

Na prethodnom fragmentu (Fragment 20) predstavljena je definicija šablona *Request*, namenjenog za prihvatanje zahteva za rezonovanje od spoljnog sistema. Kao i svaki drugi šablon, *Request* se sastoji od dva slotova (atributa koji opisuju njegovo stanje). Prvi slot je imenovan *command*, njegova podrazumevana vrednost je *nil* (nula vrednost) i ovaj slot je obavezan string tipa. Drugi slot predstavlja listu (*multislot*), imenovanu kao *params*, čija podrazumevana vrednost i tip nije specifikovana. Na primeru ovog šablona može se opisati struktura. Ovaj šablon je namenjen podršci standardnog interfejsa za API komunikaciju *IEEE 1484.11.2-2003* [106] (poglavljje 6.3.1). Spoljni sistem instancira novi objekat *Request* šablona, postavljajući komandu koju mašina treba da zaključi kao vrednost istoimenog slotova i prosleđujući ulazne podatke kao listu parametara u multislot *params*.



Ilustracija 120: Činjenice iz senke – deljivi objekti aplikacije i mašine za rezonovanje

Šabloni mogu da se definišu i na osnovu objekata iz spoljašnjeg sistema. Ovaj koncept je nazvan *činjenice iz senke* (*shadow facts*) [106] (Ilustracija 121). To su objekti koji imaju dvojnu prirodu. Za spoljnu aplikaciju, to su instance neke klase, koja je eksplicitno definisana u aplikaciji. Za ekspertski sistem, to su instance šablona (*template-a*) koji preuzimaju osnovna svojstva ovih klasa. *Činjenice iz senke* su brokerski objekti posredstvom kojih se ostvaruje komunikacija okruženja i mašine za rezonovanje. Naziv klase postaje naziv šablona, a svi atributi – podaci klase preuzimaju se kao slotovi šablona. To znači da naziv i tip atributa, postaje naziv i tip slotova šablona.

```
(import rule_reasoning.shadow.*)
(deftemplate Learner (declare (from-class Learner)))
(deftemplate Score (declare (from-class Score)))
(deftemplate Group (declare (from-class Group)))
(deftemplate Organization (declare (from-class Organization)))
(deftemplate Navigation (declare (from-class Navigation)))
(deftemplate Threshold (declare (from-class Threshold)))
```

Fragment 21: Definisanje činjenica - senki

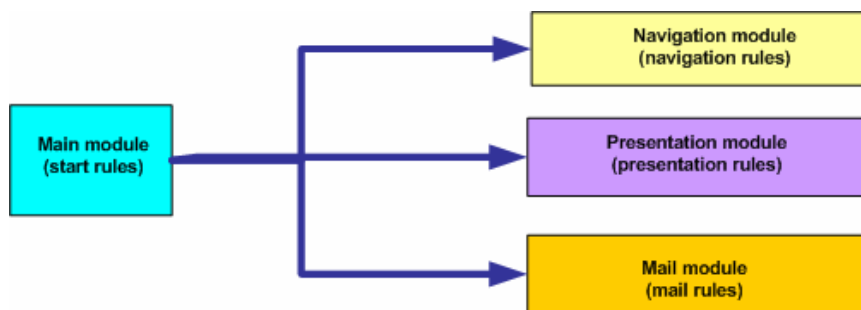
Na prethodnom fragmentu (Fragment 21), predstavljeni je šest šablona, definisanih kao činjenice iz senke. Najpre se specificira paket u kome se nalaze definicije ciljnih aplikacionih klasa, a zatim se koriste specifični oblici *deftemplate* klauzula, koji koriste refleksivnost, kao osobinu OOP da se na osnovu deklaracija klasa generišu šabloni. Činjenice iz senke su praktično interfejsi koji koriste referenciranje na zajednički resurs u memoriji, kako bi se objektno organizovano razmenjivali podaci između mašine za zaključivanje i entiteta spoljnje aplikacije – korisnika.

6.2.5 Baza pravila

Osnova za adaptaciju su odluke (zaključci) koje donosi mašina za rezonovanje na osnovu pravila kroz koja se procesiraju ulazni podaci. Platforma za rezonovanje je dizajnirana kao ekspertski sistem (u daljem tekstu ES), koji koristi *Jess* [106] shell kao platformu za rezonovanje. Produkciona pravila, zajedno sa šablonima (*templates*), čine bazu znanja. Da bi se optimizovalo korišćenje memorije i postigao što brži odziv sistema na zahteve korisnika, opredelilo se za modularan dizajn baze pravila. Pravila za adaptaciju su grupisana u tri modula (Ilustracija 122):

- Startna pravila – prihvataju zahteve za rezonovanje i aktiviraju odgovarajuću grupu pravila
- Navigaciona pravila – namenjena donošenju zaključaka vezanih za rešavanje adaptivne navigacije
- Prezentaciona pravila – namenjena donošenju zaključaka vezanih za adaptivno komponovanje sadržaja (prezentaciju).

Pored navedenih, u sistemu se koriste i pravila kojima korisnici mogu da podese filtriranje elektronske pošte (ova pravila nisu predmet razmatranja jer nemaju nikakvu vezu sa adaptacijom sadržaja učenja).



Ilustracija 121: Definisane grupe (tipovi) pravila

Modularnim pristupom u dizajnu baze pravila smanjena je i mogućnost interferencije pravila (u jednom trenutku, koriste se pravila samo jednog modula). Obzirom da su pravila u sistemu koncipirana kao objekti, koji imaju svoj naziv, prioritet, metodu koja kao ulazni parametar ima struktuiranu premisu, a kao izlazni parametar – konsekvencu, modularnošću se rešavaju i problemi rezolucije imenovanja (mogu da postoje istoimena pravila u različitim modulima).

Moduli pravila imaju unifikovanu strukturu. Sastoje se od potrebnih šablona (*templates*), ulančanih radnih i ciljnih pravila. Vrsta ulančavanja u lancu zaključivanja može da se eksplicitno podesi (u napred – *forward chaining*, ili u nazad – *backward chaining*) [108]. Radna pravila omogućavaju da se ulazi podaci prihvate, obrade, dajući međurezultate rezonovanja. Ciljna pravila za ulazne vrednosti imaju međurezultate rezonovanja, a kao konsekvencu daju konačne zaključke. Izvršenjem bilo kojeg ciljnog pravila iz modula, rezonovanje se završava.

```
(defrule start_nav (declare(salience 1000))
  (Request{command == "check_learner_nav"}) => (focus NAVIGATION))
(defrule start_prsnt (declare(salience 1000))
  (Request{command == "check_learner_prsnt"}) => (focus PRESENTATION))
```

Fragment 22: Definisanje startnih pravila

Prethodni fragment (Fragment 22) predstavlja deo glavnog modula – modula sa startnim pravilima. Rezonovanje započinje aktivacijom i izvršenjem jednog od startnih pravila. Ova pravila imaju najviši prioritet (*salience*), a pripadanjem glavnom modulu (*MAIN*), rešeno je da se prilikom instanciranja mašine za zaključivanje, automatski (podrazumevano) unose u radnu memoriju. Ova pravila prihvataju zahteve iz spoljne aplikacije (klijenta) i na osnovu zahteva aktivira jedan od modula baze znanja.

Ova funkcionalnosti se realizuje tako što se u premisama svih startnih pravila ispituje stanje slotu *command* u šablonu *Request*. U zavisnosti od naziva komande, izvršava se jedno od pravila, tako što u konsekvenci prebacuje fokus na neki od modula. To znači da se memorija za rezonovanje prazni i puni pravilima iz modula, specifikovanom u konsekvenci izvršenog startnog pravila. Na prethodnom primeru, prvo pravilo aktivira modul za rezonovanje o navigaciji, a drugo pravilo aktivira modul s pravilima za rezonovanje o prezentaciji (komponovanju sadržaja).

Kao što je prethodno objašnjeno, moduli (izuzev *MAIN* modula) se sastoje od radnih i ciljnih pravila. U ovim pravilima, pored činjenica iz senki, koriste se interne instance *Request* šablona. U njima se čuvaju međurezultati rezonovanja i preko njih se ostvaruje prenošenje istih između pravila. Na taj način se vrši propagacija zaključivanja.

6.2.5.1 Primer toka zaključivanja za rešavanje adaptivne navigacije

Na sledećem primeru je fragment skripta pravila *Navigation* modula – namenjena rešavanju adaptivne navigacije. Struktura radnih pravila (uokvirena crveno) je takođe raslojena. Najpre se izvršava pravilo (imenovano *check_learner_nav*) koje proverava da li su kroz instancu šablona *Request* (multislot *params*) prosleđeni svi potrebni ulazni parametri – identifikator predmetnog činioca i studenta (*target_org* i *leraner_id*). Pored toga, ovo pravilo pronalazi u radnoj memoriji instancu studenta koja sadrži zahtevani ID. Iz nje uzima podatak o nastavnoj grupi kojoj student pripada, a zatim u radnu memoriju

dodaje novu *Request* instancu sa novom komandom (slot *command*) i novim, kompletnijim parametrima (multislot *params*). Koncept multislota u šablonu *Request* je iskorišćen kao veoma fleksibilno rešenje za prosleđivanje parametara (međurezultata rezonovanja) bilo kog tipa i bilo kog broja.

Pošto su uslovi u premisi ispunjeni, aktivira se sledeće pravilo – *find_superorg*, koje u radnoj memoriji pronalazi instancu činioca čiji je identifikator *target_org* i iz nje uzima identifikator roditeljskog objekta. Ovo pravilo kreira novu *Request* instancu sa novom komandom (*find_nav_type*) dodajući u listu parametara identifikator roditeljskog objekta (činioca predmeta).

U sledećem pravilu (*find_nav_type*) se dolazi do tipa navigacije koji je primenjen na roditeljski predmetni činilac. Lanac zaključivanja se dalje grana u zavisnosti od tipa navigacije (*sequencing*, *selection*, *free_selection*) i svodi se na pronalaženje zavisnosti između činilaca na istom nivou hijerarhije. Za svaki činilac koji je preduslov za pristup ciljanom činiocu (*target_org*), dalje se ispituje da li je posmatrani student (*learner_id*) ispunio obaveze definisane pedagogijom (na primer, broj bodova, i/ili vreme provedeno na materijalu).

Ciljna pravila se aktiviraju kada interna *Request* instanca sadrži komandu *check_preconditions*. Ova pravila pronalaze u radnoj memoriji i ispituju performanse studenta (instanca *Score* – činjenica iz senke) za definisane kriterijume činioca – preduslova, enkapsulirane u instancu *Threshold* (činjenica iz senke). Konsekvenca ciljnog pravila je instanca šablona *Response* koja, kad se nađe u radnoj memoriji označava da je zaključivanje po zadatim parametrima zaključeno.



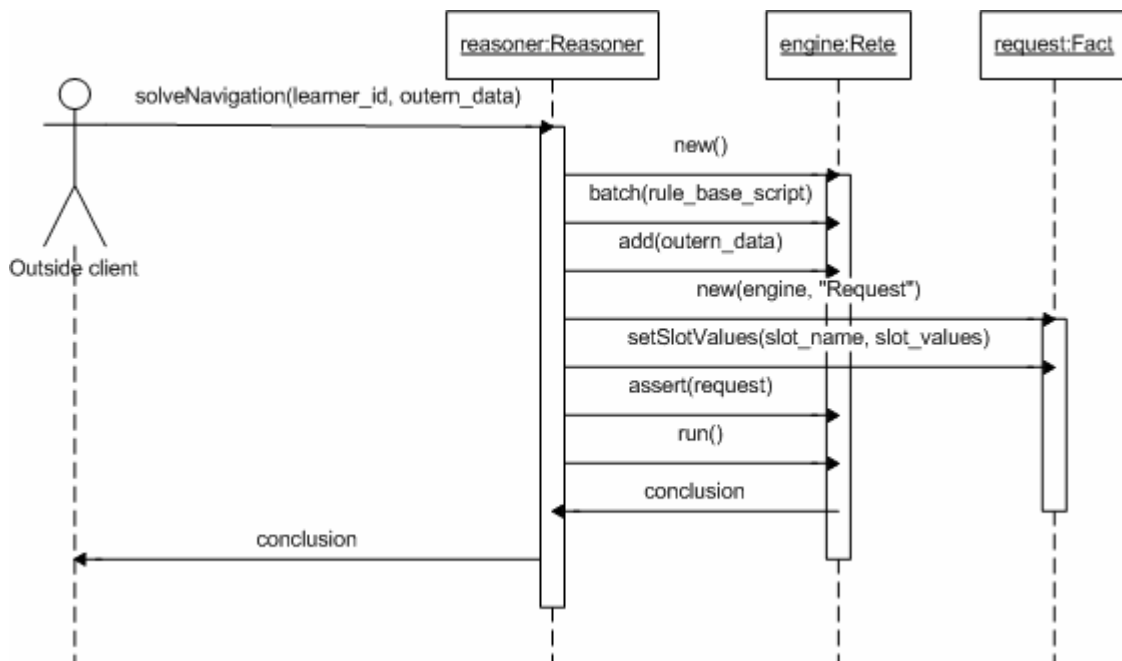
Fragment 23: Baza pravila – propagacija rezonovanja kroz pravila modula *Navigation*

U zavisnosti od vrste navigacije, koja je specificirana pedagogijom, mašina za rezonovanje se drukčije ponaša. Kod strogo sekvencijalne navigacije, činioци su povezani u jednodimenzionu ulančanu listu, tako da se proverava samo jedan činilac – preduslov. Kod slobodne navigacije, proveravaju se svi činioци – preduslovi radi sugestija koje ukazuju studentu na eventualne uzroke rezultata ispod očekivanja. To znači da se vrši ponavljanje rezonovanja, dok god u radnoj memoriji mašine za zaključivanje postoje instance činilaca – preduslova (posle zaključivanja o jednom činioци, pravila koja su izvršena se automatski izbacuju iz memorije, mašina se bez resetovanja stanja ponovo pokreće, a uslov pokretanja je stanje *Response* objekta).

Ako se radi o vođenoj selektivnoj navigaciji, ako se zaključi da uslovi nisu ispunjeni, studentu se ne dozvoljava da pristupi zahtevanom činioци, a prikazuje se odgovarajuća poruka – obrazloženje ove zabrane, sa uputstvom kako da reši problem.

6.2.6 Dinamika zaključivanja

Za uspešan rad mašine za rezonovanje, pored dobro dizajnirane baze pravila, neophodno je i ostvarivanje sinhronizovanosti sa klijentskom aplikacijom. Sinhronizacija se kontroliše iz klijentske aplikacije, poštovanjem odgovarajućeg redosleda funkcionalnih poziva.



Ilustracija 122: Sekvencijalni dijagram procesa zaključivanja u mašini za rezonovanje

Instanca klase *Reasoner* (poglavlje 6.1.2.1), prihvata spoljni zahtev preko metode *solveNavigation* zajedno sa parametrima: ID studenta o kome se rezonuje, i kolekcija podataka koji neophodni za rezonovanje (npr. nastavni činioci, rezultati i sl.). Propagacija funkcionalnog poziva se odvija u strogo definisanom redosledu. Najpre se instancira objekat klase *Rete*. Ova instanca zatim učitava skript baze znanja (*batch* metoda). Nakon toga se dodaju prosledjene činjenice (objekti) za rezonovanje (*add* metoda). Radna memorija je nakon poslednja dva funkcionalna poziva napunjena svim potrebnim podacima. Sledi formiranje činjenice (*Request*) koja sadrži naziv komande sa parametrima (koja će da inicira rezonovanje) i njeno dodavanje u radnu memoriju. Započinjanje rezonovanja je eksplicitno (*run* metod). Rezultat rezonovanja (*conclusion*) se vraća u sinhronom odzivu.

6.2.6.1 Instanciranje novih činjenica iz klijentske aplikacije

Zahvaljujući činjenicama iz senke (poglavlje 6.2.3), komunikacija između aplikacije i mašine za rezonovanje se odvija na objektnom nivou. Pored činjenica iz senke postoji još jedan komunikacioni mehanizam. Ako klijent aplikacija zna deklaracije internih šablona (*templates*) mašine za rezonovanje, ona može da instancira nove činjenice u njenoj radnoj memoriji. Sledeći kod (Fragment 24) prikazuje instanciranje šablona *Request* na osnovu deklaracije iz baze pravila (vidi prethodni sekvencijalni dijagram).

```

Fact f1=new Fact("Request",this.engine);
f1.setSlotValue("command",new Value("check_learner_nav",RU.STRING));
ValueVector vv = new ValueVector();
vv.add(new Value(org_id,RU.STRING));
vv.add(new Value(learner_id,RU.STRING));
f1.setSlotValue("params",new Value(vv, RU.LIST));
this.engine.assertFact(f1);
  
```

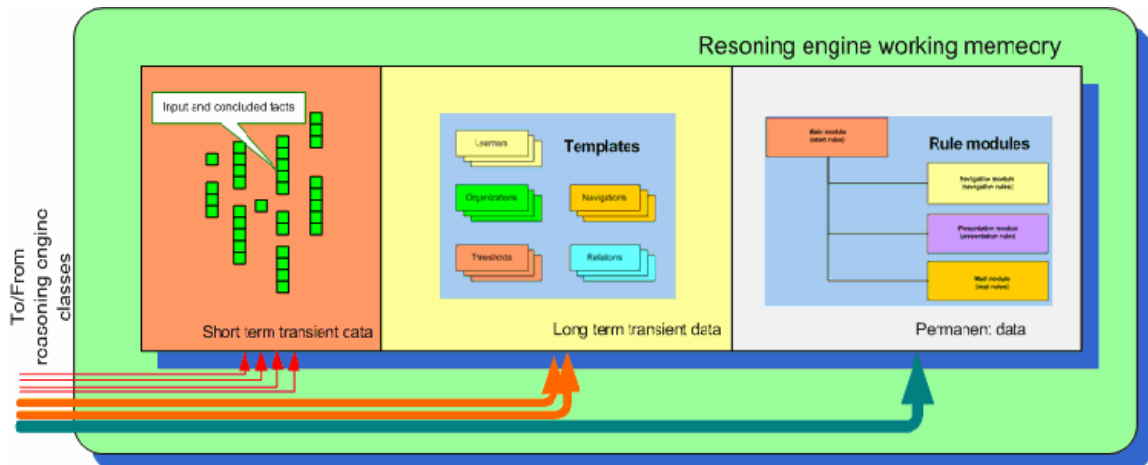
Fragment 24: Komunikacija između aplikacije i mašine za rezonovanje

Nakon učitavanja skript fajla baze pravila, korišćenjem klase *Fact* (koja enkapsulira činjenice u radnoj memoriji mašine za rezonovanje), aplikacija može da instancira šablon (*template*) kao činjenicu. Posredstvom klase *Value* dodaje vrednost slotu *command*, a posredstvom klase *ValueVector* dodaje vrednosti multislotu *params*. Nakon punjenja, šablon se ubacuje u radnu memoriju mašine za rezonovanje (metoda *assertFact*).

6.2.6.2 Kontrola radne memorije mašine za zaključivanje

Podatke (objekte) u radnoj memoriji mašine za rezonovanje prema trajanju možemo grupisati u tri celine (Slika 124):

- Permanentni podaci – pravila za zaključivanje
- Dugotrajni tranzijentni podaci – šabloni i činjenice iz senke
- Kratkotrajni tranzijentni podaci – ulazni, izlazni podaci i međurezultati rezonovanja



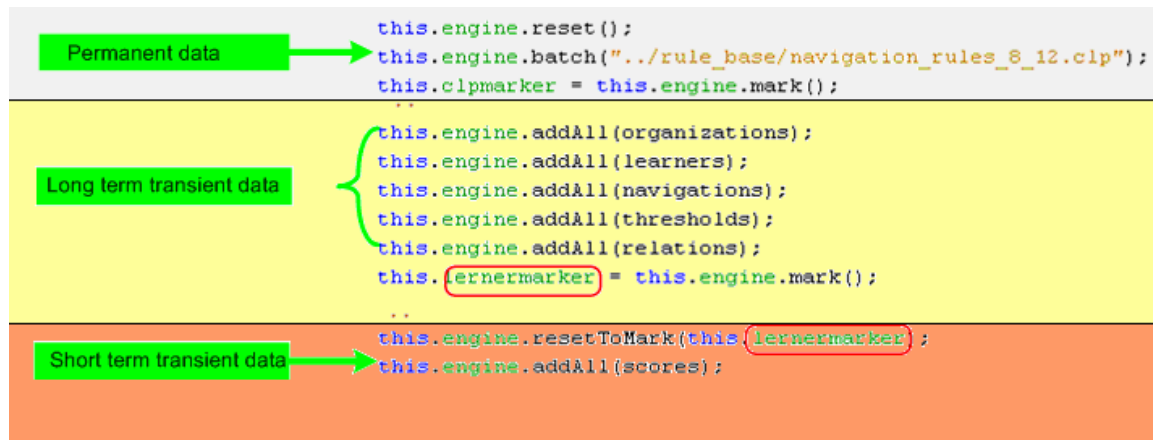
Ilustracija 123: Organizacija radne memorije mašine za rezonovanje

Na osnovu ove podele, može da se izvrši optimizacija korišćenja memorije. Ako se podaci unose u redosledu prema trajanju, memorija može da se kontroliše programatično. Sledeći kod prikazuje implementaciju kontrole memorije (Fragment 25).

Nakon unosa pravila (permanentni podaci), postavlja se objekat – memorijski marker. Zatim se unose ostali podaci potrebni za zaključivanje. Ovi podaci su tranzijentne prirode, jer se nakon svakog procesa zaključivanja menjaju prema zahtevima klijentske aplikacije. Međutim, na osnovu eksploatacionih iskustava i ovi podaci se koriste na različite načine. Tako postoje dugotrajniji podaci (podaci predmeta, navigacioni logovi, kriterijumi odlučivanja, relacije između nastavnih činilaca, studentske grupe itd.). Ovi podaci se zadržavaju u memoriji kroz više iteracija rezonovanja. Na primer, nije potrebno da se podaci o činionicima i kriterijumima odlučivanja za studente iste nastavne grupe menjaju. Nakon unosa ovakvih podataka, klijentska aplikacija postavlja drugi memorijski marker.

Kratkoročni podaci, kao što su konkretni rezultati studenata, na kojima se vrši rezonovanje, se dodaju posle drugog markera. Ako se rezonovanje obavlja u više

iteracija, korišćenjem markera, nije neophodno potpuno pražnjenje memorije, već samo iskorišćenih podataka (metoda *resetToMark*).



Fragment 25: Programatična kontrola radne memorije mašine za rezonovanje

Nakon pražnjenja memorije, dodaju se novi podaci i može da započne novi ciklus rezonovanja. Kontrola memorije je važan faktor u dizajnu mašine za rezonovanje. Ako vreme odziva nije zadovoljavajuće, pogoršavaju se performanse sistema, i u tom slučaju rezonovanje postaje balast (**poglavlje 8**). Opisana tehnika kontrola memorije predstavlja značajno rešenje za poboljšanje performansi u slučaju servisiranja velikog broja korisnika i rezonovanja nad velikom količinom podataka⁴.

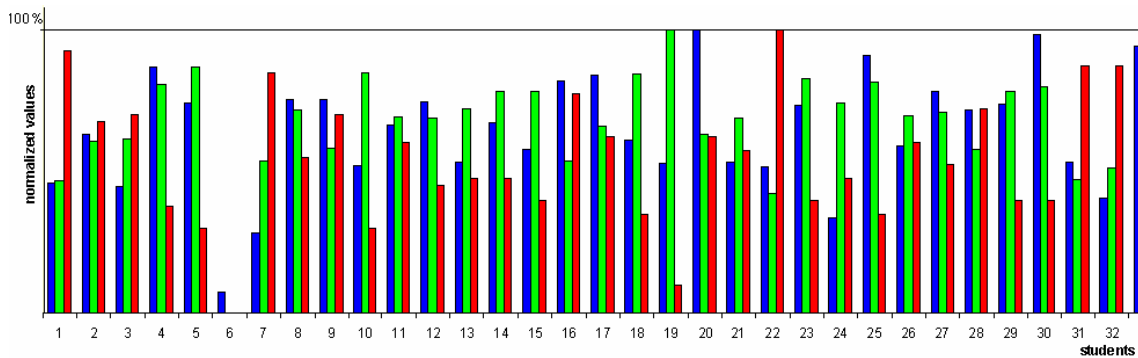
7. Evaluacija

Evaluacija predloženog rešenja obavljena je kroz tri aspekata: tehničko-tehnološki, funkcionalni i socijalni aspekt. U evaluaciji je izražen fokus na novim konceptima e-učenja, implementiranim u predloženom rešenju: na adaptivnom ocenjivanju, adaptivnoj navigaciji i sadržajima učenja. Komparativna analiza je izvršena upoređivanjem sa aktuelnim sistemima za upravljanje učenjem (LMS), pošto je predloženo rešenje najsrodnije ovoj grupi sistema za e-učenje.

7.1 Adaptivno (fuzzy) ocenjivanje

U praksi se pokazalo da klasični metodi ocenjivanja, primenjeni na testiranje posredstvom računara, ne prikazuju u potpunosti realni nivo znanja studenata. Zbog različitih uticaja okruženja (veličina prostorije, udaljenost radnim mesta, trenutne performanse računara, mrežni saobraćaj), broj tačnih odgovora nije uvek jedini parametar koji utiče na rezultat. Sledeći dijagram (Ilustracija 125) prikazuje podatke sa jedne provere (2. kolokvijum iz predmeta Računarske mreže, 2008) koja je realizovana sa 63 studenta (radi preglednosti prikazano je 32 studenta). Za svakog studenta (na x osi), prikazana su tri parametra obliku vertikalnih stubića: trajanje (plavi), broj tačnih odgovora (zeleni), broj netačnih odgovora (crveni). Sve vrednosti su normalizovane i prikazane procentualno. Na osnovu dijagrama da se zaključi da ne postoji pravilo (zavisnost) između različitih parametara koji se beleže u sistemu tokom ocenjivanja.

⁴ (aplikacija je na folderu C:\Documents and Settings\Singidunum\Desktop\Singi Doc\Doktorski rad\unis_portal_reasoning\v_1okt_08\umt_reasoning)

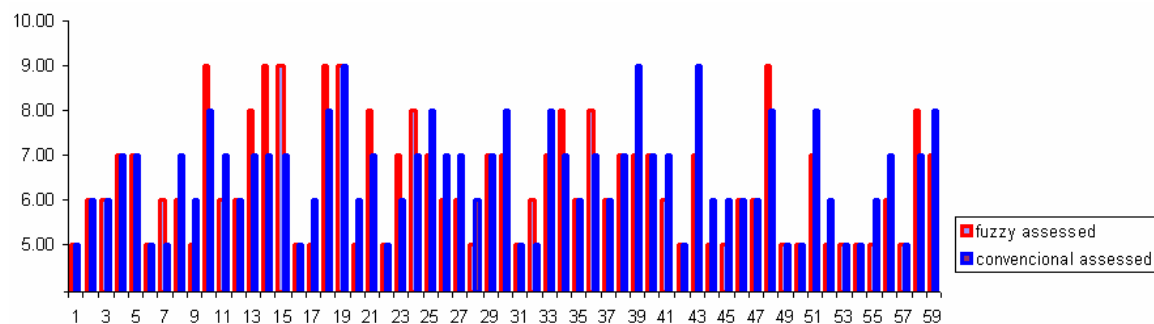


Ilustracija 124: Podaci provere (plavo – trajanje, zeleno – br.tačni odg., crveno – br.netač.odg.)

Na primer, ako uporedimo podatke studenata broj 4 i 15 može da se zaključi da, iako su oba imala podjednaki broj tačnih i netačnih odgovora, student br.4 je utrošio znatno više vremena. Kod konvencionalnog ocenjivanja (primenjeno kod gotovo svih sistema za e-učenje) ova razlika ne bi bila uzeta u razmatranje. Isti slučaj može da se uoči kod studenata br.18 i 23, ili kod studenata br.5 i 10.

Drugi karakterističan primer je da, ako se uporede studenti br.24 i 29, klasičnim ocenjivanjem drugi student će imati bolji rezultat, jer ima nešto veći broj netačnih odgovora. Ako bi se u razmatranje uzelo i vreme kao parametar, onda bi ova dva studenta imala skoro približne rezultate, jer je uticaj većeg broja netačnih odgovora kompenzovan boljim vremenskim rezultatom.

Na osnovu prethodno iznetih empirijskih razmatranja, može da se zaključi da se rezultati fuzzy i konvencionalnog ocenjivanja razlikuju. Pri uporednoj analizi (u 23 provere) fuzzy i konvencionalnog ocenjivanja (Ilustracija 126), utvrđeno je da postoji velika sličnost između rezultata u ova dva slučaja. Prosečna vrednost srednje devijacije je 0.33 (kreće se u rasponima 0 – 1, za ocene 5 – 10).

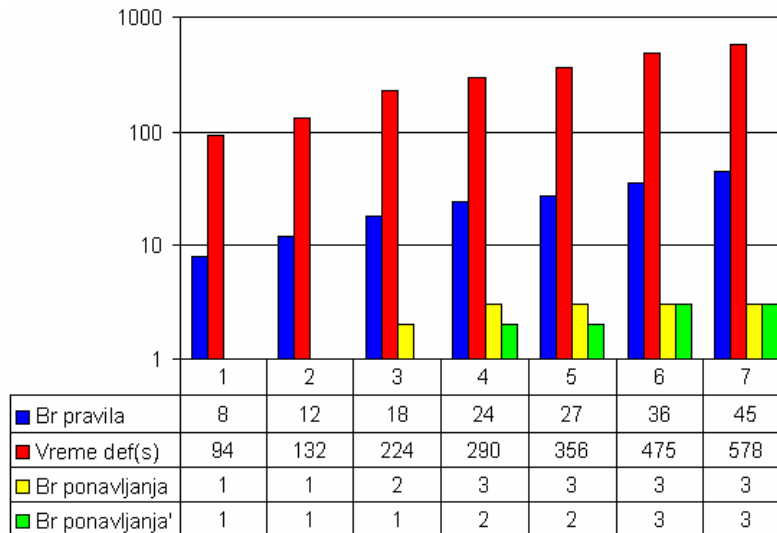


Ilustracija 125: Komparativni prikaz rezultata kod fuzzy i konvencionalnog ocenjivanja

Uprkos sličnosti, ne postoji striktna zavisnost rezultata kod fuzzy i konvencionalnog ocenjivanja. Kao što je već u radu pomenuto, adaptivno ocenjivanje (poglavlje 5.4.7) omogućava nastavnicima da na sistem prenesu svoje iskustvo u ocenjivanju provera. To znači da nastavnici samostalno uključuju kriterijume i definišu pravila za ocenjivanje. Nakon pripreme, sistem treba da ocenjuje studente na način na koji to radi nastavnik. Ako do toga ne dođe, nastavnik treba da izmeni definisane opsege termina, ili sadržaje pravila.

7.1.1 Nedostaci adaptivnog ocenjivanja

Najveća poteškoća kod adaptivnog ocenjivanja je definisanje pravila ocenjivanja od strane nastavnika. Nastavnici imaju mogućnost da definišu proizvoljan broj termina u kriterijumima za ocenjivanje. Broj pravila za ocenjivanje predstavlja proizvod broja termina u svim kriterijumima (poglavljje 5.4.7.1, izraz 17). Na sledećem dijagramu (Ilustracija 127), ilustrovan je problem koji se pojavio u toku korišćenja.



Ilustracija 126: Definisanje naprednog ocenjivanja

Izvršeno je po pet merenja za sedam slučajeva korišćenja sistema. U svim slučajevima su definisana pravila ocenjivanja za 3 ulazna kriterijuma: tačnost, netačnost odgovora i vreme odgovaranja. Slučajevi se međusobno razlikuju samo po broju definisanih termina po kriterijumu (npr. prvi slučaj ima po dva termina za svaki kriterijum, a zadnji slučaj ima 2 kriterijuma sa 3 termina i jedan sa pet termina). U praksi nije bilo slučajeva sa većim brojem termina od poslednjeg navedenog (Tabela 11).

Tabela 11: Brojevi termina po kriterijumima u slučajevima merenja

RBr	Kriterijumi			Br pravila
	Tačnost	Netačnost	Vreme	
1	2	2	2	8
2	2	2	3	12
3	3	2	3	18
4	3	2	4	24
5	3	3	3	27
6	3	3	4	36
7	3	3	5	45

Merenjima je utvrđeno da je priraštaj vremena potrebnog za definisanje pravila, veći od priraštaja broja pravila (dijagram je logaritamski, za osnovu 10), što ukazuje na kompleksnost korišćenja. Drugi pokazatelj ovog nedostatka je potreba da se izvrše korekcije pravila nakon pokušaja njihovog korišćenja (podatak *Br.ponavljanja*).

Nastavnici uspešno rukuju pravilima ako ih ima manje od 20. U praksi se najčešće koriste sistemi sa više od 20 pravila (termina). Vreme za definisanje pravila je delimično

smanjeno (time i kompleksnost sistema) automatskim generisanjem matrice pravila, u kojoj nastavnici samo treba da odaberu vrednosti izlaznog kriterijuma ocenjivanja (poglavlje 5.4.7.4).

Dodavanjem evaluatora fuzzy baze znanja (poglavlje 5.4.5.3 i 5.4.6.2) broj ponavljanja (korekcija pravila) je smanjen za jedan (*Br.ponavljanja*). Pokazalo se da evaluator ne može da pomogne u smanjivanju vremena potrebnog za definisanje pravila, ali je vrlo koristan u slučajevima kada se ocenjuje mali broj studenata (poglavlje 7.1.2).

7.1.2 Prednosti adaptivnog ocenjivanja

Najveća prednost adaptivnog ocenjivanja je da se rezultati studenata mere i posmatraju iz različitih aspekata (kriterijuma). Na taj način se postiže da ocene više odgovaraju realnom nivou i kvalitetu znanja studenta, nego što je to slučaj kod tradicionalnog ocenjivanja. Adaptivno ocenjivanje može da se podešava prema konkretnim zahtevima:

- Definisanje kriterijuma i termina ocenjivanja za svaku proveru posebno (npr. uvođenje različitih nivoa proveravanja za isti predmetni činilac – osnovni i napredni nivo).
- Uključivanje/isključivanje kriterijuma iz provere (npr. da se razmatraju samo tačnosti i vreme odgovora, ili samo tačnost i netačnost)
- Usklađivanje sistema ocenjivanja prema specifikaciji za konkretni obrazovni sistem (npr.promena pravilnika i/ili kriterijuma ocenjivanja, promena načina bodovanja i sl.)

U slučaju da se da postoji samo jedan kriterijum za ocenjivanje (npr. broj tačnih odgovora) i ako se opsezi unutar tog termina poklapaju sa opsezima za konvencionalno ocenjivanje ugrađenim u sistem (nepromenjivi, tzv.*hard*-kodirani), adaptivno ocenjivanje ima iste efekte kao konvencionalno ocenjivanje.

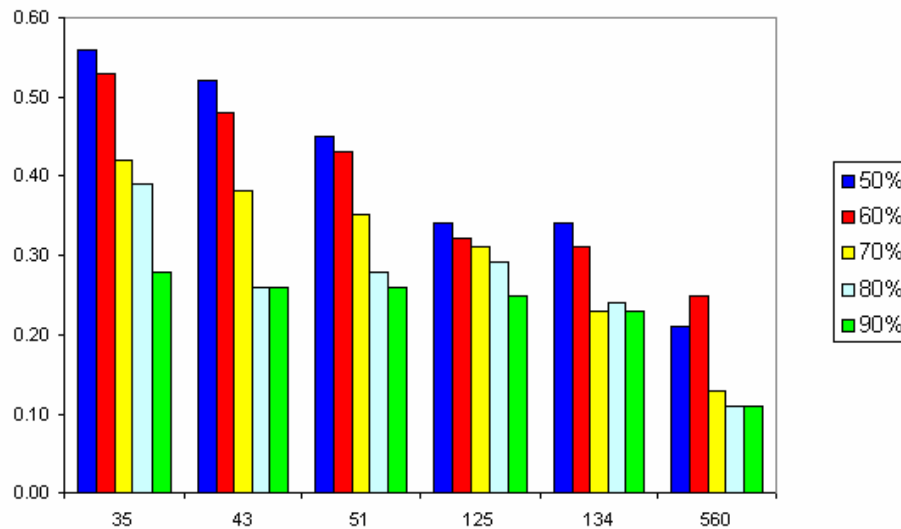
Adaptivno ocenjivanje je pogodno kod testova koji se rade posredstvom mreže. Svi rezultati korisnika se pamte i razmatraju, bez obzira da li je test završen, ili prekinut. Studenti se tada ocenjuju na osnovu odgovora do momenta prekida sesije. Adaptivnim ocenjivanjem obuhvaćeno je više kriterijuma, tako da je uključen veći broj podataka u ocenjivanje. Većim brojem pitanja i kompletnijim testom, zaključena ocena treba da više odgovara slučaju kada je student regularno završio test.

Tabela 12: Rezultati eksperimenta sa

Ukupan br.pitanja	kompletnost%											
	50		60		70		80		90		100	
	prosec.ocena	sr.devijacija	prosec.ocena	sr.devijacija	prosec.ocena	sr.devijacija	prosec.ocena	sr.devijacija	prosec.ocena	sr.devijacija	prosec.ocena	sr.devijacija
35	6.18	0.56	5.85	0.53	5.78	0.42	6.24	0.39	6.14	0.28	6.34	0.00
43	6.82	0.52	6.53	0.48	7.16	0.38	7.21	0.26	6.78	0.26	7.03	0.00
51	6.56	0.45	6.72	0.43	7.01	0.35	6.48	0.28	6.62	0.26	6.81	0.00
125	7.57	0.34	7.36	0.32	7.64	0.31	7.24	0.29	7.48	0.25	7.54	0.00
134	6.81	0.34	6.34	0.31	6.58	0.23	6.62	0.24	6.43	0.23	6.56	0.00
560	7.14	0.21	6.96	0.25	6.72	0.13	6.74	0.11	6.8	0.11	6.72	0.00

Da bi se ispitala ova tvrdnja, realizovan je eksperiment u kome su obuhvaćeni rezultati 6 provera (Tabela 12), odabranih po kriterijumu broja pitanja (cilj je da sadrže različite brojeve pitanja) i sa što većim, ali ravnomernijim brojem studenata (ispitanika). Eksperiment se sastojao od merenja prosečne ocene i njene devijacije u slučajevima kada se ocenjivalo na bazi tri kriterijuma (tačnost, netačnost i vreme odgovaranja). Svih šest provera je kompletno (svi uključeni studenti su odgovorili na sva pitanja). Na svakoj proveru izvršeno je pet merenja. U svakom merenju se varirao broj odgovora studenata, kako bi se dobila željena nekompletnost. Merene su performanse ocenjivanja za slučajeve 50% - 90% od ukupnog broja odgovora.

Da bi merenje bilo konzistentnije, studenti su dobijali odgovore jedan po jedan u nasumičnom (randomizovanom) redosledu. Na taj način, i pored nekompletnosti, davali su odgovore iz različitih nastavnih činilaca (tema, lekcija, i/ili nastavnih jedinica). U proverama je bilo uključeno prosečno 28 studenata ($\pm 10\%$). Referentni parametri su bili prosečne ocene adaptivnog ocenjivanja na kompletnim rezultatima.



Ilustracija 127: Srednje vrednosti devijacija rezultata eksperimentalnog ocenjivanja

Gornji dijagram (Ilustracija 128) vizuelno prikazuje rezultate (srednje vrednosti devijacija ocena za različite provere i različitu kompletnost) eksperimenta. Za svaku proveru (na apscisi su prikazane preko brojeva pitanja kao identifikatora: 35, 43, itd.) postoji grupa od pet stubića (*bar-ova*), a svaki predstavlja jedan slučaj merenja (sa variranjem kompletnosti odgovora: 50%, 60%, itd.). Sa povećanjem broja pitanja, odstupanje ocena nekompletnih provera od referentnih vrednosti se smanjuje. Zavisnost srednjih vrednosti devijacija od kompletnosti provere je takođe inverzna. Obzirom da adaptivno ocenjivanje ne narušava konzistentnost podataka provera, na nastavniku je opredeljenje da li da proveru ponovi, ili prizna nepotpun rezultate, ocenjujući ih na opisani način.

Tokom eksploatacije se pokazalo da je evaluator fuzzy baze znanja (poglavlje 5.4.5.3), vrlo korisna alatka za nastavnike. Iako je dizajniran prvenstveno radi provere konzistentnosti baze znanja, da ne dođe do nepravilnog rada, ili blokiranja mašine za rezonovanje, nastavnici su je koristili radi provere svojih kriterijuma na velikom broju simuliranih rezultat studenata. Preciznije, alat je eksplicitno korišćen u slučajevima kada

se proverava mali broj studenata (manje od 10). Nastavnici tada evaluiraju termine kriterijuma i pravila za ocenjivanje na velikom broju simuliranih studenata (najčešće preko sto studenata), nakon čega posmatraju simulacione ulazne podatke i odzive (ocene) i vrše fino podešavanje.

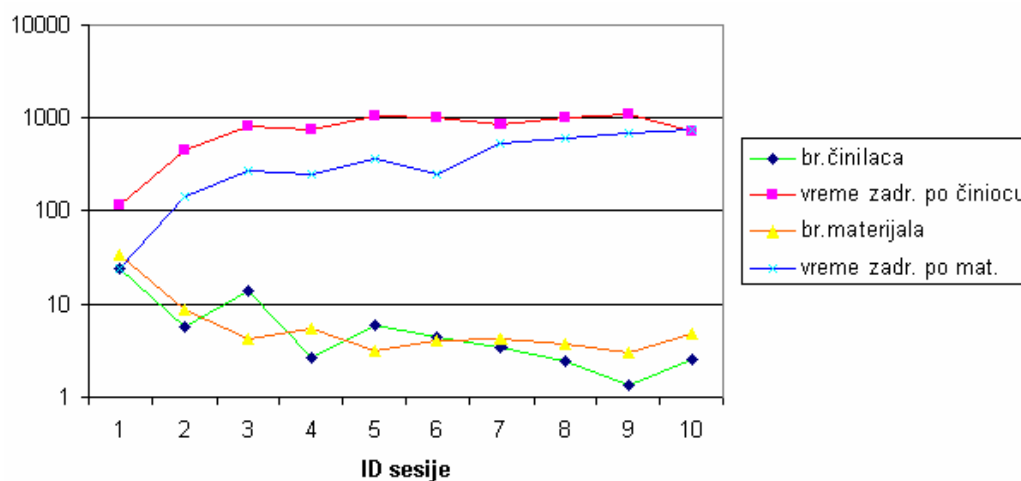
7.2 Adaptivna navigacija i sadržaji

Iskustva iz ovih segmenta adaptivnosti pokazuju potrebu za održavanjem kompleksnog studentskog modela. Kompleksnost se ogleda, ne samo u količini i vrsti podataka koje sistem prikuplja o studentima, već i u ponašanjima koje studenti ispoljavaju korišćenjem sistema. Osnovni izvor podataka za analizu navigacije i korišćenja sadržaja su tabele logova akcija studenata tokom njihovih sesija sa sistemom.

7.2.1 Adaptivna navigacija

Prvo korišćenje sistema, iako se odvijalo u kontrolisanim uslovima (u kabinetima, i uz mogućnost slobodne navigacije) i uz instrukcije i sugestije nastavnika odvijalo se na neočekivan način. Cilj prve sesije je bio da se studenti što pre upoznaju sa novim okruženjem za učenje, pre svega: da im se prezentuju sve funkcionalnosti, načini na koji pristupaju nastavnim materijalima, svrha samotestiranja, smisao anotacije sadržaja, kontekstualna sinhrona i asinhrona komunikacija.

Manji deo studenata (37%) je pratio instrukcije, a preostali studenti odmah po logovanju krenuli u samostalno *istraživanje* sadržaja koji se nude (ilustracija 137). Zbog preglednosti, prikazana je kumulativna statistika za činioce 0-tog nivoa (osnovna struktura predmeta, ili kursa). Broj činilaca i materijala kojima se pristupalo u prvoj sesiji je veliki (prosečno 34 materijala u 24 činioce), a srednje vreme zadržavanja je malo (od 10-ak sekundi do 2 min.). Ove, skoro nasumične akcije, su interpretirane kao traganje za informacijama radi obaveštenosti, ali ne i radi samog učenja. Iz tog razloga je dodata mogućnost da se na struktuirane činioce mogu dodavati resursi, koji bi pružali osnovne informacije o činioce (poglavlje 6.2.1, ilustracija 124).



Ilustracija 128: Statistika prvih 10 studentskih sesija

Nakon prve sesije, uključena je pedagogija, koju su definisali nastavnici. Pedagogija je implementirana povezivanjem činilaca u relacije, dodavanjem pitanja za samotestiranje

(na svaki činioc koji ima materijal) i definisanjem preduslova koje studenti moraju da ispune da bi pristupili činiocu. Primenjenom pedagogijom, navigacija je postala adaptivna, što znači da je sistem dinamički generisao linkove, koji su bili selektivno i sekvencijalno dostupni, u skladu sa stanjem studentskog modela za svakog individualnog studenta. Primena adaptivne navigacije je promenila sliku korišćenja sistema (sesije 3 – 10). Broj činilaca i materijala je počeo da pada i konvergira (prosečno oko 4 pristupa). Istovremeno, prosečno vreme zadržavanja imalo je tendenciju rasta i konvergencije (prosečno oko 7 minuta).

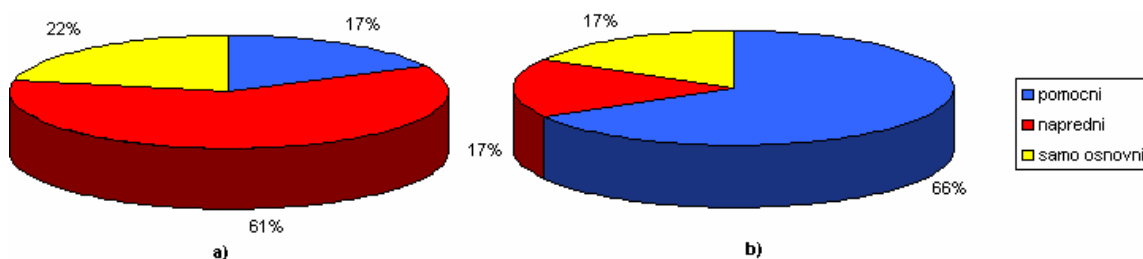
Kao zaključak analize slučaja „Prvih deset sesija“ je da adaptivna navigacija vrlo brzo rezultira vidljivim efektima:

- Studenti koriste sistem na način predviđen namenom,
- Sistem je relaksiran manjim brojem poziva od klijentske ka serverskoj strani, čime je dobijeno vreme za rad sa kritičnim resursima (npr. komponovanje i distribuirane memorijski zahtevnih sadržaja, rad mašine za rezonovanje),
- Efekti implementirane pedagogije se kroz navigaciju ispoljavaju na bolje fokusiranje, vođenje i napredovanje studenata kroz nastavu.

7.2.2 Adaptivni sadržaji

Adaptivnost sadržaja u predloženom rešenju je realizovana proširivanjem standardne strukture sadržaja po *IMS CP* (poglavljje 3.2.3) i *SCORM* (poglavljje 3.2.4) specifikacijama. Dodavanje ekstenzije je bilo neophodno, pošto sistemi za upravljanje učenjem (za koje su navedeni standardi dizajnirani) nemaju mogućnost adaptacije sadržaja. Istovremeno, težilo se da se zadrži jednostavnost i intuitivnost korišćenja nastavničkog alata za rad sa sadržajima.

Adaptivnost sadržaja je rešena njegovom kontekstualizacijom (poglavljje 6.2.3). Isti sadržaji mogu da se koriste u različitim činiocima na različite načine. Na primer u nekom činiocu, sadržaj će biti predstavljen u osnovnom kontekstu, a u drugom, isti taj sadržaj koristiće se samo kao pomoć (pomoćni kontekst) u slučaju da student nije savladao osnovni sadržaj.



Ilustracija 129: Dijagrami korišćenja sadržaja

Na prethodnom dijagramu (ilustracija 138), predstavljena je statistika korišćenja sadržaja kod 127 studenata, na 24 činioca u 2 predmeta, izabrana po kriterijumu što bolje fragmentiranosti (što manji činioci, sa sadržajima koji objašnjavaju pojedinačne koncepte predmetnog znanja) i kompletnosti (dobro definisani činioci: postoje preduslovi, pitanja za samotestiranje i sadržaji za sva tri definisana konteksta). Prvi pita dijagram (a)

prikazuje način korišćenja sadržaja studenata koji su iz prvog puta savladali sadržaje osnovnog konteksta. Najveći deo njih je pristupio sadržajima naprednog konteksta (61%), manji deo pomoćnim sadržajima (17%), a ostatak studenata je imao satisfakciju rezultatima na osnovnom nivou izučavanja (22%).

Drugi pita dijagram prikazuje distribuciju korišćenja kod studenata koji nisu iz prvog puta savladali sadržaje osnovnog konteksta (46 studenata). Najveći broj njih je, odmah posle prvog prolaska, pristupio pomoćnim materijalima (66%), a interesantno je da je jedan manji deo (17%) pokušao da pristupi naprednim materijalima.

Prikazana raspodela je ispunila očekivanja ovakvog modela struktuiranja nastavnih sadržaja. Analizirani rezultati su maksimizovani, jer je doseg posmatranja bio korišćenje malog broja ciljno izabranih sadržaja, kreiranih, fragmentiranih i opisanih na preporučen način. Ne sme da se zaboravi da mogućnosti adaptacije isključivo zavisi od načina na koji nastavnik strukturiira i prezentuje predmetno znanje. Ako su nastavni sadržaji loše fragmentirani, obimni, ako postoje preklapanja između činilaca istog predmeta i ako sadržaji nisu kontekstualizovani, ne postoje potrebni preduslovi za adaptaciju. U tom slučaju, izmerene performanse neće da zadovolje očekivanja korisnika sistema.

7.3 Upoređenje sa srodnim sistemima za e-učenje

Postoje različite komparativne analize sistema za e-učenje. Neke su fokusirane na procenjivanje upotrebljivosti sistema od strane korisnika [115]. U tom slučaju, komparacije su bazirane na anketiranju nastavnika i studenata, skaliranju odgovora ispitanika i rangiranju sistema. Najčešće su korišćene Likertova bipolarna skala (sortiranje od najgoreg do najboljeg) i Ipsativna skala (izaberi najboljeg po zadatom kriterijumu).

Druga vrsta analiza su usresređene na merenje tehničko-tehnoloških karakteristika i funkcionalnosti sistema [116], [117]. Ove analize su bazirane na upoređivanju navedenih karakteristika (na primer platformska (ne)zavisnost, (ne)zavisnost od SUBP, performanse korišćenih serverskih tehnologija i sl.) i na merenjima dodatnih svojstava (na primer, srednje vreme odziva i devijacija vremena odziva, maksimalan broj korisnika koji može da se opsluži). U analizama, kriterijumi funkcionalnosti su najčešće grupisani u kategorije:

- Mogućnosti ocenjivanja
- Mogućnosti pristupanja (korišćenja) sadržajima učenja.
- Mogućnosti uvođenja, obavezivanja i praćenja aktivnosti korisnika
- Mogućnosti komunikacije između korisnika
- Mogućnosti kreiranja i razvoja sadržaja učenja

U ovoj evaluaciji je primenjena komparativna analiza po funkcionalnostima iz dva razloga. Prvi je da nije postojala mogućnost obezbeđivanja uslova da ispitanici koriste funkcionalnosti različitih sistema za upravljanje učenjem. Drugi razlog je da postoji zavisnost funkcionalnosti sistema od tehničko-tehnoloških karakteristika.

Na sledećoj tabeli (Tabela 9) predstavljen je uporedni pregled funkcionalnosti aktuelnih sistema za upravljanje učenjem (LMS) i predloženog rešenja (*UNIS Portal*). Performanse sistema su bodovane na najjednostavniji način (1/0 – ima/nema), obzirom da je izvršena fragmentacija po više kriterijuma razvrstanih u četiri kategorije: sadržaji, studentski model, provera znanja i kolaborativnost.

Tabela 13: Uporedni prikaz karakteristika različitih sistema za upravljanje učenjem

Sistemi	Kategorije																									
	Sadržaji										Studentski model			Proveravanje znanja					Kolaborativnost							
	Tekst	Slika	Audio	Video	Import	Eksport	Kreiranje	Anotacija	Adaptivnost sadržaja	Navigaciona adaptivnost	Praćenje akcija	Praćenje navig. putanja	Praćenje vremena	Kooperativnost modela	Jednokriterijumsko	Pretest	Ispit	Vremenski ograničeno	Višekriterijumsko	Promena načina ocenjivanja	Chat	Mail	Forum	Obaveštenja	Kontekstualna komunikacija	Skor
Moodle	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1	1	1	18
Oracle i-Learning	1	1	1	1	1	1	1	0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	1	0	13
Web CT	1	1	1	1	1	1	1	0	0	1	1	0	0	0	1	1	1	1	0	0	1	1	1	1	0	15
IBM Lotus	1	1	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	14
A Tutor	1	1	1	1	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	0	15
Sakai	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1	1	0	14
dotLRN	1	1	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1	1	0	13
UNIS Portal	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	23

U svim aktuelnim LMS postoje mogućnosti za predstavljanje i distribuiranje multimedijalnih sadržaja. Razlike se odnose na personalizaciju sadržaja. Od svih sistema samo Moodle i predloženo rešenje imaju mogućnosti anotiranja sadržaja, s razlikom da Moodle pruža samo mogućnost anotacije od strane nastavnika, i korišćenje iste u upitima pri pretrazi sadržaja od strane svih zainteresovanih korisnika. U predloženom rešenju implementirane su napredne funkcionalnosti anotacije sadržaja.

Adaptivna navigacija je implementirana samo u sistemima WebCT, Sakai i A Tutor i to samo u verzijama koje podržavaju *SCORM 2004* standard, koji definiše pravila za navigaciju. Ova adaptivnost je samo delimična i određena je striktnim pravilima, podržanim IMS standardom (poglavlje 6.2). U predloženom rešenju, zahvaljujući implementaciji rezonovanja kroz ekspertski sistem i na bazi produkcionih pravila, postoji veća navigaciona adaptivnost (poglavlje 6.2 i 7.2).

Adaptivnost sadržaja nije implementirana niti u jednom od analiziranih sistema. Prost razlog je što implementirani standardi ne omogućavaju adaptivnost sadržaja, već samo njegovo struktuiranje i povezivanje za predmetne činioce. Drugi razlog je što u LMS nema rezonovanja na bazi ekspertskih sistema, već samo primitivno odlučivanje na bazi ispitivanja stanja pojedinih parametara u bazi podataka.

Svi posmatrani sistemi imaju mogućnost praćenja akcija korisnika. Obim i kompleksnost podataka koji se pamte u logovima su u direktnoj proporciji sa kompleksnošću studentskog modela. Oracle i-Learning, WebCT i A Tutor nemaju ugrađene mehanizme za praćenje navigacionih putanja. Od svih sistema, samo predloženo rešenje ima mogućnost zapisa vremena početka i završetka pojedinih akcija.

Interesantna činjenica je da niti u jednom sistemu nije eksplicitno definisan studentski model. Samo se napominju generalno korišćeni standardi. Analizom dostupnih shema BP

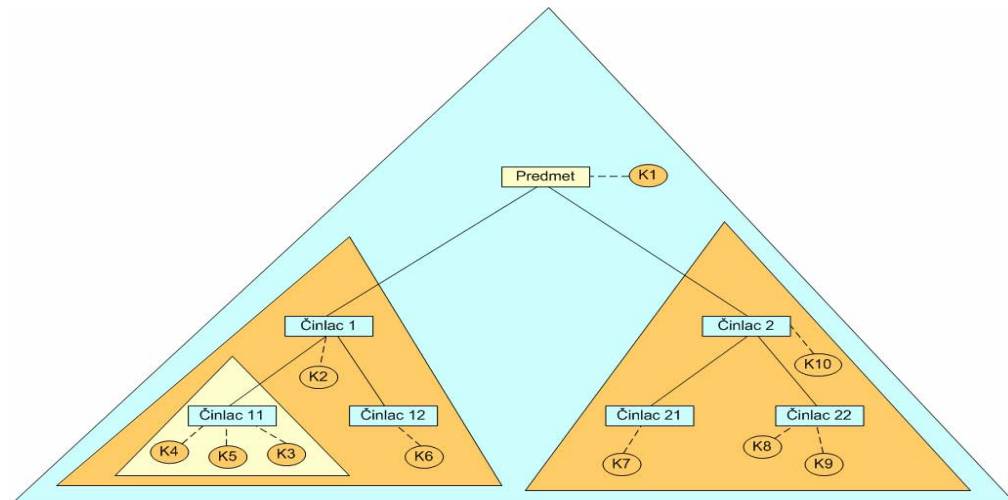
zaključak je da je kod većine sistema delimično podržan IMS LIP standard (poglavljje 3.1.2), ali niti u jednom ne postoji alat koji omogućava serijalizaciju/deserijalizaciju objekata studentskog modela. Ova činjenica ukazuje da je nemoguć uvoz ili izvoz modela na korisničkom nivou, a time niti kooperacija sa srodnim sistemima.

Svi sistemi omogućavaju jednokriterijumsko proveravanje znanja studenata kroz preteste (samotestiranje) i kroz ispite (eksplicitno definisane provere). WebCT, IBM Lotus i predloženo rešenje imaju mogućnost vremenskog ograničavanja provera. U posmatranim sistemima nema višekriterijumskog ocenjivanja i mogućnosti promene načina ocenjivanja (adaptivno ocenjivanje) izuzev u predloženom rešenju (poglavljja 5.4. i 7.1).

7.3.1 Inovativno rešenje za sinhronu komunikaciju

Kod svih poređenih sistema ugrađena je kolaborativna komponenta učenja kroz različite vidove sinhronog (chat) i asinhronog (mail, forum, obaveštenja) komuniciranja (Tabela 9). Kontekstualna kolaborativnost, koja predstavlja implementaciju konstruktivističkog pristupa učenju (poglavljje 2.1.1), implementirana je samo u dva sistema: u Moodle-u i u predloženom rešenju.

Pored navedenog, predloženo rešenje ima funkcionalnost više: sinhrona komunikacija se odvija u kontekstu predmetnih činilaca, što znači da ne moraju eksplicitno da se zadaju teme i definišu aktivnosti kroz koje se komuniciranje odvija. Ovakvo rešenje dovodi u analogiju predmetne činioce i koncept „soba“ kod takozvane *chat rooms* komunikacije. Time se postiže grupisanje korisnika koji koriste iste sadržaje učenja, kao što *chat* sobe okupljaju korisnike koji dele zajednička interesovanja.



Ilustracija 130: Dosezi vidljivosti kontekstualne komunikacije

Razlika *chat rooms* modela i kontekstualne sinhronne komunikacije u predloženom rešenju je u hijerarhijskoj struktuiranosti. Viši nivoi hijerarhije u predloženom rešenju omogućavaju veću *vidljivost* korisnika. U gornjem slučaju (ilustracija 139) korisnik K1, koji se navigaciono nalazi na vrhu predmetne hijerarhije (npr. pristupio je silabusu predmeta), videće sve druge korisnike, koji koriste resurse posmatranog predmeta (na nižim nivoima hijerarhije) i moći će da s njima stupi u komunikaciju. Korisnik K2, koji se nalazi na predmetnom činiocu br.1, videće korisnike na činiocima 11 i 12 (K4 – K6), ali neće videti nijednog korisnika na drugim predmetnim činiocima istog nivoa (npr. K7 –

K10). Primenjen koncept ispoljava za učenje korisnu fleksibilnost jer omogućava različito kontekstualno filtriranje komunikacija, a time selekciju korisnika i njihovo povezivanje po osnovu učenja.

8. Zaključak

U doktorskom radu je predstavljeno softversko rešenje (*UNISPortal*) sistema za e-učenje. Ovo rešenje je idejno i praktično potpuno novo u odnosu na postojeće srodne sisteme. Teorijska osnova rada je u prikazanim pregledima i klasifikacijama teorijskih dostignuća fenomenologije učenja. Pored toga, predstavljena su praktična dostignuća iz oblasti elektronskog učenja: standardi, arhitekture sistema, korišćene tehnologije, idejna i praktična rešenja. U predstavljenom rešenju ugrađena je respektivna količina teorijskog znanja i praktičnog iskustva u bavljenju sistemima za e-učenje. Dokazano je da se mogu realizovati sistemi koji ispoljavaju inteligentno ponašanje, kroz različite oblike adaptivnosti i individualizacije procesa učenja, a koji u isto vreme imaju podjednako izražene administrativne funkcionalnosti, svojstvene samo sistemima za upravljanje učenjem.

Naučni doprinosi

U prvom delu dokorskog rada je izvršena analiza sadržaja svih aktuelnih teorija na kojima se zasnivaju savremeni sistemi za e-učenje (poglavlje 2). Istaknuta je kompleksnost fenomena učenja, i neophodnost interdisciplinarnog pristupa, kako bi se omogućila njegova personalizacija. U analizi je predložena podela navedenih teorija na ekstrospektivne (poglavlja 2.1.1 do 2.1.4) i introspektivne (poglavlja 2.1.7 do 2.1.10). Za svaku pojedinačnu teoriju, pored opisa dat je i kritički osvrt na istu i njenu primenljivost u savremenim sistemima za e-učenje.

U cilju potpunog sagledavanja kompleksnosti fenomena učenja, pored navedenih teorija, prikazani su aktuelni stilovi učenja (poglavlje 2.1.5) i nastavni stilovi (poglavlje 2.1.6). Predstavljene su različite kategorizacije stilova učenja kao načina spoznaje stvarnosti. Dat je kritički osvrt na stilove učenja da oni definišu čiste stereotipe kao apstraktne definicije. Čisti stereotipi u stvarnosti ne postoje. Kod realnih studenata oni se uzajamno prepliću, a koncept stereotipa je fabrikovan radi diferenciranja stilova. Nastavni stilovi su izvedeni iz stilova učenja. Da bi sistem bio adaptivan i da bi se učenje personalizovalo, nastavni stilovi treba da predstave odgovor na svaki poseban stil učenja.

Sistemi za e-učenje ispoljavaju inteligentno ponašanje ako imaju sposobnost adaptacije resursa učenja svakom pojedinačnom studentu. Postoje dva osnovna preduslova za adaptaciju: postojanje modela studenta i postojanje modela za struktuiranje i organizovanje resursa učenja. U radu su kritički predstavljeni svi aktuelni standardi za modelovanje studenata (poglavlje 3.1) i modelovanje resursa učenja (poglavlje 3.2).

Ukazano je na veliku sličnost standardnih studentskih modela, kod kojih su razlike u načinu grupisanja pojedinih podataka. S druge strane je istaknuto da su u adaptivnim sistemima za e-učenje razvijani specifični studentski modeli, vanstandardni, fokusirani samo na konkretan problem učenja koji se rešava. Zaključak je da ne postoji univerzalan studentski model, ali da postoje nekoliko standardnih inicijativa kao što su *IEEE PAPI* (poglavlje 3.1.1) i *IMS LIP* (poglavlje 3.1.2) čije preporuke treba da se ugrade u vlastita rešenja. Navedeni standardi omogućavaju proširenja (za adaptaciju u zavisnosti od

specifičnosti primene) i obuhvataju sve osnove parametre studenta, neophodne za eventualnu razmenu (saradnju) između različitih edukativnih sistema.

Osnovna ideja standarda za modelovanje resursa učenja je da omogući višestruko korišćenje istih (*reusability*) i implementaciju naprednih mehanizama pretrage. Za razliku od standarda za modelovanje studenata, ovi standardi su kompatibilni i noviji predstavljaju proširenja prethodnih. Na primer, *DublinCore – dc* (poglavljje 3.2.1) je preteča svih ostalih. *LOM* standard (poglavljje 3.2.2) se pojavio kao potreba za proširenjem opisa (metapodataka) resursa učenja. Na osnovu *LOM* standarda dizajnirana je paleta *IMS* standarda (pre svega *IMS LR*) koji dodaju mogućnost pakovanja i organizovanja resursa u kompleksnije strukture i kreiranje testova (poglavljje 3.2.3). Danas možda najviše zastupljen – *SCORM* standard (poglavljje 3.2.4), obuhvata u sebi sve prethodne, dodajući nove koncepte, kao što su različiti tipovi resursa i koncept aktivnosti. Navedeni standardi omogućavaju delimično ostvarivanje adaptivnosti sistema – delimičnu navigacionu adaptivnost. Adaptacija sadržaja nije specifikovana. Iz tog razloga, ovi standardi su primenjeni u predloženom rešenju, ali su im dodata neophodna proširenja.

Nakon kratkog pregleda razvoja sistema za e-učenje (poglavljje 3.3), opisane su pojedine vrste sistema za e-učenje na osnovu opšte prihvaćene klasifikacije na adaptivne i neadaptivne sisteme (poglavljja 3.4 i 3.5). Naučni značaj ovog dela je da su u svakom pojedinačnom opisu predstavljena arhitekturna rešenja, funkcionalnosti, prednosti i nedostaci u odnosu na druge, srodne sisteme.

U prikazu predloženog rešenja (poglavljje 4), predstavljena je nova, preciznija kategorizacija alata za podršku elektronskom učenju, izvršena na osnovu dva kriterijuma: funkcionalnost i korisnički tipovi. Predstavljene kategorije alata, iako različite po funkcionalnostima, preklapaju se kod različitih korisničkih tipovima. Ovo svojstvo ne sme da se zanemari u metodologiji razvoja sistema za e-učenje, jer omogućava optimizaciju u fazama dizajna i implementacije, i jednostavnije održavanje i administriranje aplikacije u fazi korišćenja.

U radu je predstavljen novi koncept – adaptivno ocenjivanje (poglavljje 5.4). Adaptivnost se ogleda u selektivnom izboru kriterijuma po kojima se vrednuje znanje studenata, i u mogućnosti da se skaliranje i vrste ocena prilagode konkretnim edukativnim sistemima. Da bi se obezbedila veća upotrebljivost sistema i ocenjivanje na osnovu delimičnih, nekompletnih podataka, primenjeno je neegzaktno rezonovanje na bazi fuzzy logike. Alat za dizajn naprednog ocenjivanja omogućava nastavnicima da, bez poznavanja fuzzy logike, korišćenjem prirodnog jezika prenesu svoje iskustvo u ocenjivanju na sistem. Time ponašanje sistema u ocenjivanju prestaje da bude rigidno, već podražava način na koji to rade nastavnici koji su kreirali kriterijume i pravila ocenjivanja.

Precizno je definisana ontologija fuzzy koncepata, potrebna za rezonovanje (poglavljje 5.4.5.1). Formalizovan je metodološki pristup fuzzy rezonovanju, po kome se ono odvija po tačno utvrđenim fazama (5.4.6.1). Radi provere konzistentnosti celokupnog sistema, dizajniran je poseban modul za evaluaciju (5.4.6.2), koji omogućava da se proverí kompletna baza fuzzy znanja, na osnovu simuliranih podataka korisnika. Pored navedenog, izvršena je kritička analiza prednosti i nedostataka adaptivnog ocenjivanja primenjenog u sistemu (poglavljje 7.1).

Za rešavanje adaptivne navigacije i sadržaja (poglavljje 6.2) korišćen je ekspertski sistem, koji radi na bazi produkcionih pravila i diskretne logike. Ovaj deo primenjene inteligencije eksploatiše zaključke rezonovanja na bazi fuzzy logike (u vidu diskretnih numeričkih vrednosti, dobijenih defazifikacijom rezultujućih fuzzy skupova), koristeći ih kao ulazne vrednosti u sistemima pravila. Zaključivanjem se dobijaju odluke koje se koriste za adaptivnu navigaciju i komponovanje sadržaja.

Predstavljeno je arhitekturalno rešenje (poglavljje 6.2.2) i proširena je struktura sadržaja učenja data IMS i SCORM specifikacijom (poglavljje 6.2.3). Na taj način omogućena je nova klasifikacija nastavnih materijala, prema kontekstu u kojima materijal može da se koristi. Korelacijom stanja studentskog modela sa kontekstom nastavnog materijala u okviru posmatranog nastavnog činioca, omogućena je adaptacija sadržaja i kod domenski nezavisnih sistema (u predloženom rešenju).

U radu je formalizovana struktura baze pravila, potrebna za adaptivnu navigaciju i komponovanje sadržaja (poglavljje 6.2.5). Pravila su grupisana u module prema nameni, a pojedini moduli se aktiviraju u zavisnosti od konkretnog korisničkog zahteva. U svakom modulu, pravila su grupisana na tri celine: startna, radna i ciljna pravila, bez obzira na svrhu rezonovanja. Obzirom da se predloženo rešenje koristi kao serverska aplikacija, u uslovima izvršavanja više klijentskih procesa, predstavljen je način optimizovanog upravljanja radnom memorijom mašine za rezonovanje (poglavljje 6.2.6.2). Svi podaci se raspoređuju u tri memorijska dela. Pravila se čuvaju u tzv. permanentnoj memoriji, šabloni i činjenice iz senke u dugoročnoj tranzijentnoj memoriji, a ulazni, izlazni podaci i međurezultati se čuvaju u kratkoročnoj tranzijentnoj memoriji.

Poseban doprinos predstavlja evaluacija predloženog rešenja (poglavljje 7), sa fokusom na ispitivanje različitih adaptivnih svojstava. Za analizu adaptivnog ocenjivanja analizirana su iskustva iz eksploatacije (nedostaci) i rezultati eksperimenta nad postojećim, realnim podacima proverena (poglavljje 7.1.2). Evaluacija adaptivne navigacije i sadržaja (poglavljje 7.2) izvršena je kroz statističku analizu logova aktivnosti studenata prilikom korišćenja sistema. Primenjeni koncepti su rezultirali boljim fokusiranjem, vođenjem i napredovanjem studenata kroz nastavu. Kroz sve segmente evaluacije, prožima se zaključak da, ukoliko ima više implementirane pedagogije, sistem ispoljava inteligentnije ponašanje (veću adaptivnost i bolju personalizaciju resursa učenja), a odziv studenata i njihovi rezultati su time bolji.

Praktični doprinosi

Primenom u radu opisanih standarda e-učenja, omogućena je dekompozicija arhitekture predloženog rešenja na nezavisne i međusobno slabo spregnute module (poglavljje 4.1). U radu su korišćena originalna rešenja, kojima je sistem jasno raslojen na klijentski i serverski deo, koji komuniciraju posredstvom servisnog međusloja. Ovaj sloj čine softverske komponente – davaoci servisa. Korišćenjem RIA tehnologija, aplikaciona logika je podeljena između klijentske i serverske strane (poglavljje 4.2). Kod aktuelnih sistema za upravljanje učenjem je sva logika izuzev validacije podataka korisničkih unosa, na serverskoj strani. Pomenuta tehnologija je omogućila kreiranje bogatog korisničkog interfejsa i logike na klijentskoj strani. Servisni međusloj je doprineo poboljšanju fleksibilnosti i skalabilnosti serverske strane sistema. Komponente su u potpunosti zamenjive, a čitav sistem lako proširljiv novim modulima.

Za potrebe naprednog ocenjivanja dizajniran je originalni vizuelni alat za definisanje termina kriterijuma za ocenjivanje (poglavljje 5.4.7.3). Isto tako, dizajniran je vizuelni alat za povezivanje predmetnih činilaca u relacije zavisnosti, a koje predstavljaju jedan od osnovnih preduslova adaptivne navigacije (poglavljje 5.5). Glavni motiv vizuelizacije je da se omogući nastavnicima da što jednostavnije i intuitivnije kreiraju sadržaje pedagogije, neophodne za adaptivnost i personalizaciju procesa učenja.

U radu je predstavljena je nova dimenzija kolaborativne funkcionalnosti – implicitna kontekstualna komunikacija (poglavljje 6.1.2.3 i 7.3.1). Ovaj koncept je nov, ali teorijsku osnovu ima u konstruktivističkoj teoriji učenja. Ova teorija, iako vremesna, aktuelizovana je u gotov svim savremenim sistemima za e-učenje. Prikazano rešenje korisniku nudi tip komunikacije i korisnike u zavisnosti od konteksta kome je pristupio posredstvom navigacije. Ako je kontekst prijava ispita, nudi se asinhrona komunikacija sa personalom studentske službe. Ako student koristi nastavni materijal, sistem nudi sinhronu komunikaciju sa svim trenutno dostupnim korisnicima koji su pristupili ovom, ili drugim resursima istog predmetnog činioca. Navigacijom u dubinu predmetne strukture, komunikacija se sve više fokusira, a korisnici se filtriraju na kontekstualnom nivou.

Predložen način hijerarhijskog povezivanja (vidljivosti) korisnika nikada ne može da naruši okvire edukativnog konteksta. S druge strane, korisnik uvek može da proširi komunikaciono okruženje napuštajući detaljne i pristupajući predmetnim činiocima višeg hijerarhijskog nivoa.

U svim aktuelnim sistemima za e-učenje, komunikacija između korisnika se deklarativno predstavlja kao deo konteksta učenja (poglavljje 6.1.3). Međutim, kroz praktične implementacije to nije dovoljno izraženo. Uveden je koncept aktivnosti, koji se koristi za eksplicitnu kontekstualizaciju, jer se aktivnosti vezuju za konkretne nastavne činioce. Nedostatak koncepta aktivnosti je da je njeno definisanje ekskluzivno pravo nastavnika. To znači da je potrebno da se za čitav nastavni proces definišu aktivnosti, kako bi se unutar njega sačuvala kontekstulana komunikacija. Ako to nije slučaj, komunikacija nije vezana za kontekst učenja.

Predstavljeno rešenje sadrži alat za anotaciju sadržaja, koji omogućava kontekstualnu kolaborativnost. Kontekstualni koncept je dizajniran na bazi foruma. Forumi kao vid asinhrona kontekstualne (tematske) komunikacije, omogućavaju ravnopravan položaj svih učesnika. Primenom koncepta foruma, daje se mogućnost ravnopravnog rada svih subjekata nastavnog procesa. Teme mogu da zadaju i studenti, ne samo nastavnici. Forumi su međutim i dalje posebna struktura u odnosu na organizacionu strukturu predmetnih činilaca, što znači da konteksti ne moraju da se podudaraju sa kontekstima predmetnih činilaca.

U predloženom alatu za anotaciju, studenti mogu da dodaju anotacije na svaki pojedinačni nastavni materijal. One mogu biti privatne, kao vlastite beleške i javne, kao komentari. Ako student proglasi anotaciju javnom, ona postaje vidljiva za sve korisnike. Isto tako, vidljivi su i svi ostali publikovani komentari. Na ovaj način, anotirani sadržaj, kao kod foruma, postaje tematska nit za kontekstualno povezivanje anotacija različitih korisnika. Anotacije pored sadržaja, poseduju i podatke kao što su naslov, vreme objavljivanja i ime kreatora, tako da mogu da se pretražuju po različitim kriterijumima. Fuzijom koncepata anotacija i foruma, u predloženom rešenju je dobijen novi kvalitet

kolaborativnog učenja. Omogućena je implicitna kontekstualna asinhrona komunikacija i filtriranje vidljivosti korisnika i informacija vezanih za konkretan edukativni kontekst.

Pravci daljeg razvoja

Težište daljeg razvoja sistema biće na proširivanju funkcionalnosti na nove sadržaje i tipove učenja. Jedan od pravaca biće dizajn modula za učenje socijalnih veština, kao što je interpersonalna komunikacija u različitim poslovnim kontekstima i na različitim jezicima. Novi modul bi obuhvatao alat za kodiranje domenskog znanja nastavnika (npr. iz bankarske, marketinške, berzanske i druge vrste službene korespodencije), alat za građenje toka komunikacije iz elementarnih komunikacionih fragmenata (npr. tipa izjava – komentar, ili pitanje - odgovor), komponentu za praćenje i adaptaciju toka komunikacije sistema i studenta i komponentu za lokalizaciju u ciljni jezik.

Drugi pravac razvoja biće omogućavanje učenja rešavanjem problema (*problem based learning*). Ovaj način učenja, karakterističan za deo inteligentnih tutorijskih sistema, naročito je pogodan za razvoj veština rešavanja problema iz oblasti primenjene matematike u ekonomiji, bankarstvu, finansijama, osiguranju, elektronici, računarskim mrežama, programiranju i drugim egzaktnim oblastima u kojima se traži veština prepoznavanja tipa problema i poznavanja procedura za njihovo rešavanje. Ovaj modul bi obuhvatao alat za kodiranje domenskog znanja (kroz sistem pravila i/ili ograničenja), generator problema i rešenja, i komponentu za adaptaciju – korigovanje tipa i sadržaja problema u skladu sa ispoljenim greškama studenta.

Oba predložena pravca daljeg razvoja upotpunjavaju adaptivna svojstva predloženog rešenja i omogućavaju veći stepen specijalizacije, koji je inače zahtevan na višim nivoima edukacije studenata.

9. Literatura

- [1] D.C. Phillips i Jonas F. Soltis, "Perspectives on Learning", knjiga *Teachers College Press*, 1984, poglavlje 3.
- [2] About learning, http://www.funderstanding.com/about_learning.cfm
- [3] R. M. Felder, L. K. Silverman: "Learning and Teaching Styles in Engineering Education". *Engineering Education* 78,7, (1988) strana 674-681.
- [4] Keefe, J.W., "Learning Style: An Overview," in Keefe, J.W., ed., *Student Learning Styles: Diagnosing and Prescribing Programs*, Reston, Va.: National Association of Secondary School Principals, 1979
- [5] Lawrence, G., *People Types and Tiger Stripes: A Practical Guide to Learning Styles*, 3rd ed., Gainesville, Fla.: Center for Applications of Psychological Type, 1993.
- [6] Felder, R.M., Felder, G.N., and Dietz, E.J., "The Effects of Personality Type on Engineering Student Performance and Attitudes," *Journal of Engineering Education*, Vol. 91, No. 1, 2002, pp. 3–17. Online at <http://www.ncsu.edu/felder-public/Papers/longmbti.pdf>
- [7] Felder, R.M., "Reaching the Second Tier: Learning and Teaching Styles in College Science Education," *Journal of College Science Teaching*, Vol. 23, No. 5, 1993, pp. 286–290.
- [8] Stice, J.E., "Using Kolb's Learning Cycle to Improve Student Learning," *Engineering Education*, Vol. 77, No. 5, 1987, pp. 291–296.
- [9] Kolb, D.A., *Experiential Learning: Experience as the Source of Learning and Development*, Englewood Cliffs, N.J.: Prentice-Hall, 1984.
- [10] Learning Theories of Instructional Design, <http://www.usask.ca/education/coursework/802papers/mergel/brenda.htm>
- [11] Benay Phyllis Dara-Abrams, *Applying Multi-Intelligent Adaptive Hypermedia to Online Learning*, Union Institute & University Graduate College, Ph.D. , March 2002
- [12] Richard M. Felder: "Learning and Teaching Styles in Engineering Education (Author's Preface – June 2002)" <http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Papers/LS-1988.pdf>
- [13] Kobsa, A., Wahlster, W., *User Models in Dialog Systems*, Springer-Verlag, 1989
- [14] Kobsa, A., Koenemann, J. i Pohl, W., "Personalized Hypermedia Presentation techniques for Improving Online Customer Relationships". *The Knowledge Engineering Review*, 2001, 16(2), str. 111-155.
- [15] ELM-ART: adaptivni hipermedijalni sistem za učenje LISP programskog jezika, posećen 2008, na sajtu <http://apsymac33.uni-trier.de:8080/Lisp-Course>

- [16] Kobsa, A., Miller, D., and Nill, A. KN-AHS: An Adaptive Hypertext Client of the User Modeling System BGP-MS. *Proceedings of the Fourth International Conference on User Modeling*, Hyannis, 1994, MA, str. 99-105.
- [17] Encarnacao, L.M., „Multi-level user support through adaptive hypermedia: a highly application-independent help component“, *International Conference on Intelligent User Interfaces*, 1997, ACM, NY USA, str. 187-194.
- [18] Ontologging konzorcijum: "User modeling issues in the context of Knowledge Management Systems", Project ID: IST-2000-28293, 2000, www.ontologging.com
- [19] IEEE PAPI Learner – model studenta, http://edutool.com/papi/papi_learner_07_main.pdf
- [20] Technical Committee Workgroup 2 – dostupan na sajtu: <http://www.jtc1.org/>
- [21] Dolog, P., Nejdl, W.: "Challenges and Benefits of the Semantic Web for User Modeling," *In Proceedings of AH2003 Workshop at 12th World Wide Web Conference*, Budapest, Hungary, May 2003
- [22] IMS LIP specifikacija, dostupna na sajtu <http://www.imsglobal.org>
- [23] C. Smythe, F. Tansey, and R. Robson, "IMS Learner Information Packaging Information Model Specification," Public Draft Specification. [Online]. Available at: <http://www.edtech.vt.edu/edtech/ims/LIPPDSEInfoModelv0p4.pdf>
- [24] Dolog, P., Nejdl, W.: "Challenges and Benefits of the Semantic Web for User Modelling", dostupno na <http://www.wis.win.tue.nl/ah2003/proceedings/um-1/>, 2003
- [25] Internet2 inicijativa, dostupna na <http://middleware.internet2.edu>
- [26] EDUCAUSE inicijativa, dostupna na <http://www.educause.edu/eduperson/>
- [27] Lindholm, J., "Metadata resources – Metadata standards and specifications for describing people and their interests", 2002. dostupno na <http://www.ukoln.ac.uk>
- [28] EduPerson Object Klass Specification (200604), Internet2 Middleware Architecture Committee for Education, Directory Working Group (MACE-Dir), 2006, Specifikacija dostupna na <http://www.nmi-edit.org/eduPerson/internet2-mace-dir-eduperson-200406.html>.
- [29] Mealling, M., "A URN Namespace of Object Identifiers", IETF RFC 3061, Februar 2001. Specifikacija dostupna na <http://www.ietf.org/rfc/rfc3061.txt>
- [30] E. Maler et al. "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) ". OASIS, Septembar 2003. ID dokumenta oasis-sstcsaml-core-1.1. Specifikacija dostupna na <http://www.oasis-open.org/committees/security/>.

- [31] *ELENA* projekat, informacije dostupne na sajtu <http://www.elena-project.org/>
- [32] *ELENA* projekat, OPIS, dostupan na sajtu istraživačke organizacije *ISTWORLD*: <http://www.ist-world.org/ProjectDetails.aspx?ProjectId=1fbecfe5302c47a0989f6d1c919d0863>
- [33] "Challenges and Benefits of the Semantic Web for User Modelling"
- [34] Dublin Core Metadata Initiative, osnovni podaci se mogu naći na sajtu <http://dublincore.org/>
- [35] Dublin Core Metadata Initiative rečnik tipova *DCMITYPE*, dostupan na <http://dublincore.org/documents/dcmi-type-vocabulary/>
- [36] Rečnik geografskih naziva: *Thesaurus of Geographic Names – TGN*, dostupan na sajtu <http://www.getty.edu/research/tools/vocabulary/tgn/index.html>
- [37] Format datuma tipova *W3CDTF* razvijen za standard ISO 8601, dostupan na sajtu <http://www.w3.org/TR/NOTE-datetime>
- [38] Rečnik medijalnih tipova na Internetu - *MIME*, dostupan na sajtu: <http://www.iana.org/assignments/media-types/>
- [39] Rečnik za opis svetskih jezika - *RFC4646* , dostupan na sajtu: <http://www.ietf.org/rfc/rfc4646.txt>
- [40] A. Steinacker, A. Ghavam, and R. Steinmetz, "Metadata Standards for Web-Based Resources", IEEE Multimedia, January-March 2001, pp. 70-76.
- [41] Specifikacija proširenja LOM standarda za Veliku Britaniju – *UK LOM CORE*, dostupna na sajtu: <http://www.cetis.ac.uk/profiles/uklomcore>
- [42] Specifikacija proširenja LOM standarda za Kanadu – *Can Core*, dostupna na sajtu: <http://www.cancore.ca/>
- [43] Specifikacija proširenja LOM standarda za Australiju – *Vetadata*, dostupna na sajtu: <http://e-standards.flexiblelearning.net.au/vetadata/index.htm>
- [44] Specifikacija proširenja LOM standarda za Norvešku – *NORLOM*, dostupna na sajtu: <http://www.nssl.no/>
- [45] Bailey, W. „What Is IMS Simple Sequencing?“, Centre For Educational Technology Interoperability Standards – *CETIS*, dostupno na sajtu <http://www.cetis.ac.uk/static/briefings.html>
- [46] Uhr, L. "Teaching machine programs that generate problems as a function of interaction with students" Materijal sa *The 24th National Conference*, 1969, str. 125-134.
- [47] Suppes, P. "Some theoretical models for mathematics learning" *Journal of Research and Development in Education*, Br.1, 1967, str.5-22.
- [48] Greeno, J. G., Collins, A., Beranek, B. i Resnick, L. B. "Cognition and Learning". U knjizi *Handbook of educational psychology*, 1994, str. 1-51.

- [49] Sleeman, D. i Brown, J. S. "Introduction: Intelligent Tutoring Systems". Iz knjige istoimenih autora *Intelligent Tutoring Systems*, 1982, str. 1-11.
- [50] Brown, J. S. i Burton, R. R. "Diagnostic models for procedural bugs in basic mathematical skills". Časopis *Cognitive Science*, Br.2, 1978, str.155-192.
- [51] Anderson, J. R. "The Architecture of Cognition", izdanje od *Cambridge, Massachusetts: Harvard University Press*, 1983.
- [52] Rosenberg, R. "A Critical Analysis of Research on Intelligent Tutoring Systems". Časopis *Educational Technology*, Br.27(11), 1987, str.7-13.
- [53] Wenger, E. (1987). "Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge". Izdanje od Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1987.
- [54] Sack, W., Soloway, E., i Weingrad, P. "Re-Writing Cartesian Student Models". U knjizi J. E. Greer i G. I. McCalla, *Student Modelling: The Key to Individualized Knowledge-Based*, Berlin: Springer-Verlag 1994, str. 355-376.
- [55] P. Brusilovsky, E. Schwartz, G. Weber, "ELM-ART: An Intelligent Tutoring System on the World Wide Web", Materijal sa treće međunarodne konferencije o inteligentnim tutorskim sistemima, Montreal, Kanada, 1996, str. 261-269.
- [56] Joseph Beck, Mia Stern, i Erik Haugsjaa, "Applications of AI in Education", 2000, dostupno na sajtu: <http://www.acm.org/crossroads/xrds3-1/aied.html>
- [57] S. Ritter, "PAT Online: A Model-Tracing Tutor on the World-Wide Web", Materijal sa Workshop-a "Intelligent Educational Systems on the World Wide Web", Kobe, Japan, August 1997, str. 11-17.
- [58] Cognitive Tutor, ITS za učenje matematike, dostupan na sajtu: <http://www.carnegielearning.com>
- [59] A. Mitrović, K. Hausler, "Porting SQL-Tutor to the Web", Materijal sa *International Workshop on Adaptive and Intelligent Web-based Educational Systems*, Montreal, Kanada, 2000, str. 50-60.
- [60] SQL Tutor , ITS za učenje SQL jezika, dostupan na sajtu: <http://www.cosc.canterbury.ac.nz/~tanja/sql-tut.html>
- [61] NORMIT (2003), <http://www.cosc.canterbury.ac.nz/~tanja/normit.html>
- [62] KERMIT (2003), <http://www.cosc.canterbury.ac.nz/~tanja/kermit.html>
- [63] CAPIT (2003), <http://www.cosc.canterbury.ac.nz/~tanja/capit.html>
- [64] G. Weber, P. Brusilovsky, "ELM-ART: An Adaptive Versatile System for Web-Based Instruction", *International Journal of Artificial Intelligence and Education*, Vol.12, 2001, str. 351-384.
- [65] ELMART, AEHS sistem, dostupan na sajtu: <http://www.psihologuie.univ-trier.de:8000/project/ELM/elmart.html>

- [66] Brusilovsky, P. "Adaptive Educational Systems on the World Wide Web". Materijal iz *The World Congress on Expert Systems Workshop on Current Trends and Applications of Artificial Intelligence in Education*. Mexico City, 1998, str. 9-16.
- [67] Kircher, M., Prashant, J. "Pooling Pattern". *European Conference on Pattern Languages of Programs- EuroPLoP* 2002. Dostupno na: http://www.hillside.net/patterns/EuroPLoP2002/papers/Kircher_Jain.zip.
- [68] L.A. Zadeh. „Fuzzy sets“. *Information and Control*, 8 str. 338-383; 1965.
- [69] Du,X., Zhang,N., Ying,H. „Structure analysis and system design for a class of Mamdani fuzzy controllers“, *International Journal of General Systems*, Vol. 37, No. 1, Taylor&Francis, februar 2008, str.83–101.
- [70] L.H. Tsoukalas and R.E. Uhrig. *Fuzzy and Neural Approaches in Engineering*. John Wiley & Sons Inc. New York 1996.
- [71] Materijal „Basic Fuzzy Mathematics for Fuzzy Control and Modeling“, dostupan na sajtu: http://media.wiley.com/product_data/excerpt/73/07803349/0780334973.pdf
- [72] Web stranica „Kronecker proizvod“, dostupna na sajtu: http://en.wikipedia.org/wiki/Kronecker_product
- [73] Lewis, H.W., „The Foundations of Fuzzy Control“, New York: Plenum Press, 1997.
- [74] Tong, R.M., Synthesis of fuzzy models for industrial processes—some recent results. *International Journal of General Systems*, 4(3), 1978, str. 143–162.
- [75] Skale ocenjivanja u različitim visokobrazovnim sistemima, dostupne na sajtu: [http://en.wikipedia.org/wiki/Grade_\(education\)](http://en.wikipedia.org/wiki/Grade_(education))
- [76] Jovanović, J., Gašević, D. & Devedžić, V., „Dynamic Assembly of Personalized Learning Content on the Semantic Web“, *3rd European Semantic Web Conference*, Crna Gora, 2006, str. 545-559.
- [77] IMS SS standard, dostupan na sajtu: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bestv1p0.html
- [78] Perez-Marin, D., Alfonseca, E., Rodriguez, P. & Pascual-Nieto, I. „Automatic Generation of Students' Conceptual Models from Answers in Plain Text“, *11th International Conference on User Modeling - UM 2007*, Greece, Jul 2007, Springer-Verlag, str.329-333.
- [79] Reload editor – aplikacija za SCORM pakovanje kurseva, dostupna na sajtu: <http://www.reload.co.uk>
- [80] eXe-learning – aplikacija za SCORM pakovanje kurseva, dostupna na sajtu: <http://exelearning.org>
- [81] Docyrus – aplikacija za SCORM pakovanje kurseva, dostupna na sajtu: <http://www.docyrus.net>

- [82] QuizCreator – aplikacija za izradu kvizova, dostupna na sajtu: <http://www.sameshow.com/quiz-creator.html>
- [83] Quiz Builder – aplikacija za izradu kvizova, dostupna na sajtu: <http://www.quiz-builder.com/>
- [84] Hot Potatoes – aplikacija za izradu kvizova, dostupna na sajtu: <http://hotpot.uvic.ca/>
- [85] CTAT – skup autorskih alata za ITS Cognitive tutor, dostupan na sajtu: <http://ctat.pact.cs.cmu.edu/>
- [86] ToolBook – autorski alat za Docent LMS, dostupan na sajtu: <http://www.toolbook.org/>
- [87] Adobe Authorware – autorski alat po SCORM standardu, dostupan na sajtu: <http://www.adobe.com/products/authorware/>
- [88] Lectora – autorski alat po SCORM standardu, dostupan na sajtu: <http://trivantis.com/products/lectora.html>
- [89] Java programski jezik, dostupan na sajtu: <http://www.sun.com/java/>
- [90] WebORB servis provajder, dostupan na sajtu: <http://www.themidnightcoders.com/weborb/>
- [91] PHP tehnologija (programski jezik) za Web aplikacije, dostupna na sajtu: <http://www.php.net/>
- [92] JSP tehnologija za Web aplikacije, dostupna na sajtu: <https://java.sun.com/products/jsp>
- [93] Http Servlet tehnologija za Web aplikacije, dostupna na sajtu: <http://java.sun.com/products/servlet/>
- [94] ASP tehnologija za Web aplikacije, dostupna na sajtu: <http://www.asp.net/>
- [95] Python programski jezik i tehnologija za Web aplikacije, dostupan na sajtu: <http://www.python.org/>
- [96] AJAX tehnologija za Web aplikacije, dostupna na sajtu: <http://www.ajax-community.de/projekte/4307-ajax-homepage.html>
- [97] Spring framework, dostupan na sajtu: <http://www.springframework.org/>
- [98] Struts framework, dostupan na sajtu: <http://struts.apache.org/>
- [99] JSF framework, dostupan na sajtu: <http://www-jlc.kek.jp/subg/offl/jsf/>
- [100] JavaScript skript jezik za Web aplikacije, dostupan na sajtu: <http://javascript.about.com/>
- [101] ActionScript skript jezik za Web aplikacije, dostupan na sajtu: <http://www.actionscript.org/>
- [102] Adobe Flex 3 tehnologija, dostupan na sajtu: <http://www.adobe.com/products/flex/>

- [103] MySQL sistem za upravljanje relacionom bazom podataka, dostupan na sajtu: <http://www.mysql.com/>
- [104] Pregled performansi SUBP, dostupan na sajtu: http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems
- [105] Šimić, G., Devedžić, V. "Building an intelligent system using modern Internet technologies", *International Journal Expert Systems with Applications*, Elsevier, Vol.25, 2003, str. 231-246.
- [106] Friedman-Hill, E. "Jess In Action", Manning Publications Co.,2003, str. 264-283
- [107] AICC CMI data model – retrieved May 19, 2008 from <http://www.aicc.org/>
- [108] JESS – Java Expert System Shell, dostupan na sajtu: <http://herzberg.ca.sandia.gov/>
- [109] Šimić, G. "The Multi-course Tutoring System Design", *International Journal of Computer Science and Information Systems*, Vol.1, 2004, str.141-155
- [110] Šimić, G (2007). UNIS.Multitutor – Multimedia presentation, retrieved January 30, 2008, from <http://people.singidunum.ac.yu/index.php?ID=gshimic>
- [111] PHP – Java Bridge – tehnologija za povezivanje PHP Web stranica sa Java klasama, dostupna na sajtu: <http://php-java-bridge.sourceforge.net/pjb/>
- [112] Zimmermann, C, *FuzzySet Theory and Its Applications*, Second Edition, Kluwer Academic Publishers, 1990
- [113] Larman, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd edition, Prentice-Hall, 2005.
- [114] Draw2D biblioteka za izgradnju bogatih klijentskih aplikacija, dostupna na sajtu: <http://www.draw2d.org/>
- [115] Instructional Technology Resource Center, „LMS Focus Group Report“, *Idaho State University*, 2006, dostupan na sajtu: <http://elearning.fe.up.pt/docs/lms-focus-group-report.pdf/view>
- [116] Savčić, J. „Avatars and 3d Virtual World on Learning“, The 5th International Conference on Informatics, Educational Technology and New Media in Education, Serbia, 2008., str.91-97.
- [117] *LMS Options and Comparisons*, dokument dostupan na sajtu: http://kumu.brocku.ca/webct/LMS_Options_and_Comparisons
- [118] ADL - Advanced Distributed Learning Initiative, dostupan na sajtu: <http://www.adlnet.gov/>
- [119] Aimeur, E., Salehian, B. "A multimedia training system applied to telephony", *The 3rd IEEE International Conference on Advanced Learning Technologies*, 2003,str.272 - 273.

- [120] Virvou, M. Tsiriga, V., "Web Passive Voice Tutor: an intelligent computer assisted language learning system over the WWW", *IEEE International Conference on Advanced Learning Technologies*, 2001, str.131-134.
- [121] Tsiriga, V.; Virvou, M., "Initializing the student model using stereotypes and machine learning", ", *IEEE International Conference on Advanced Learning Technologies*, 2002, str.404 - 409.
- [122] Brown, J. S., Burton, R. R., & deKleer, J. „Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III“, U editorskoj knjizi D. Sleeman & J. S. Brown, *Intelligent Tutoring Systems*, str. 227-282.
- [123] Brusilovsky, P., A Distributed Architecture for Adaptive and Intelligent Learning Management Systems, *AIED 2003 Workshop Towards Intelligent Learning Management Systems*, Sydney, 2003, str. 5-13.
- [124] InterBook, adaptivni hipermedijalni sistem, dostupan na sajtu: <http://www.contrib.andrew.cmu.edu/~plb/InterBook.html>
- [125] Moodle, sistem za upravljanje učenjem, dosupan na sajtu: <http://moodle.org/>
- [126] Blackboard, sistem za upravljanje učenjem, dosupan na sajtu: <http://www.blackboard.com/>
- [127] ATutor, sistem za upravljanje učenjem, dosupan na sajtu: <http://www.atutor.ca/view/2/1665/1.html>
- [128] Olat, sistem za upravljanje učenjem, dosupan na sajtu: <http://www.olat.org/>
- [129] Sakai projekat, sistem za upravljanje učenjem, dosupan na sajtu: <http://sakaiproject.org/>
- [130] Dokeos, sistem za upravljanje učenjem, dosupan na sajtu: <http://www.dokeos.com/>
- [131] Desire2learn, sistem za upravljanje učenjem, dosupan na sajtu: www.desire2learn.com
- [132] Mikut,R., Burmeister,O., Groll,L. ,Reischl,M., "Takagi-Sugeno-Kang Fuzzy Classifiers for a Special Class of Time-Varying Systems", *IEEE Transactions on Fuzzy Systems*, Vol.4, 2008, str.1038 - 1049.
- [133] Shoureshi, R., Zhi Hu, "Tsukamoto-type neural fuzzy inference network", *American Control Conference*, Vol.4, 2000, str.2463 – 2467.
- [134] Alat za dizajn fuzzy baze znanja – *fuzzyJToolkit*, dostupan na sajtu: http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit.html
- [135] CLIPS – jezik i alat za izgradnju ekspertskih sistema, dostupan na sajtu: <http://clipsrules.sourceforge.net/>
- [136] Materijal „Frame Based Expert Systems“, dostupan na sajtu: <http://web.njit.edu/~theo/courses/cis370/Lecture06.pdf>

10. Skraćenice

AEHS – Adaptive Educational Hypermedia System (adaptivni edukativni hipermedijalni)
AH – Adaptive Hypermedia (adaptivni hipermedijumi)
AEH – Adaptive Educational Hypermedia (adaptivni edukativni hipermedijumi)
AHS – Adaptive Hypermedia System (adaptivni hipermedijalni sistem)
API – Application Programming Interface
BP – baza podataka
DB – Data Base (vidi BP)
DBMS – Database Management System (vidi SUBP)
DC – Dublin Core (specifikacija za opis sadržaja učenja)
ES – Expert System
IEEE – Institute of Electrical and Electronics Engineers
IEEE PAPI – IEEE Public and Private Information (vidi IEEE)
IMS – Interchange Modelling Specification
IMS LIP – IMS Learner Information Package (vidi IMS)
IMS QTI – IMS Question and Test Interoperability (vidi IMS)
IMS SS – IMS Simple Sequencing
IS – informacioni sistem
ITS – Intelligent Tutoring System (inteligentni tutorski sistem)
JESS – Java Expert System Shell
LMS – Learning Management System (sistem za upravljanje učenjem)
LO – Learning Object (objekat učenja)
LOM – Learning Object Metadata (specifikacija za opis sadržaja učenja)
OOP – Object Oriented Programming
RIA – Rich Interface Application
RDBMS – Relational DBMS (vidi DBMS)
SCO – Sharable Content Object (deljivi sadržaj učenja)
SCORM – Sharable Content Object Reference Model (specifikacija za opis i referenciranje deljivih sadržaja)
SK – slučaj korišćenja
SQL – Structured Query Language (struktuiran jezik za upite nad BP – vidi BP)
SUBP – sistem za upravljanje BP (vidi BP)
UC – Use Case (vidi SK)
UML – Unified Modelling Language
UNIS – University Network Information System