# Sistem za Interaktivnu Obuku i Testiranje Znanja iz Algoritama i Struktura Podataka

Miloš Milivojević, Đorđe Đurđević i Milo Tomašević

***Sadržaj* — U radu se predlaže novi softverski sistem za samostalnu interaktivnu obuku i proveru znanja studenata iz oblasti algoritama i struktura podataka. Sistem predstavlja nadgradnju Vizuelnog Simulatora Algoritama (VSA), ranije razvijenog na Elektrotehničkom fakultetu u Beogradu. Sistem omogućava kontrolu nivoa samostalnosti obučavanog lica, podržava fleksibilno zadavanje skupa ulaznih podataka i konfigurabilno automatsko ocenjivanje testa na osnovu postupka izvršenog od strane testiranog lica. U radu se opisuje arhitektura predloženog sistema i daju relevantni detalji implementacije.**

***Ključne reči* — automatsko ocenjivanje, interaktivna obuka, provera znanja.**

## I. Uvod

Softverski sistemi za obuku i proveru znanja predstavljaju značajna sredstva u procesu obrazovanja. Sa jedne strane, oni omogućavaju ponavljanje lekcije (odnosno postupka obuke) u meri u kojoj je to potrebno obučavanom licu. Sa druge strane, nakon završenog ciklusa obuke, oni omogućavaju proveru stečenog znanja, uz eventualno automatsko ocenjivanje. Tokom obe navedene aktivnosti, primena ovih sistema smanjuje prosečno vreme koje instruktor mora da posveti obučavanim licima (odnosno licima koja se testiraju) do mere u kojoj prisustvo instruktora nije neophodno, što dozvoljava instruktoru da više vremena odvoji na aktivnosti koje zahtevaju veću pažnju. Takođe, napredniji sistemi omogućavaju obradu rezultata testiranja, čijom analizom instruktor dobija indikatore na osnovu kojih može da prilagodi sadržaj lekcija i testova.

Najzastupljeniji sistemi za obuku i proveru znanja – tzv. *virtuelna okruženja za učenje* (eng. *virtual learning environments*) [1] – baziraju se na pitanjima sa ponuđenim odgovorima ili pitanjima na koja se odgovor daje unosom teksta. Kod ovakvog načina davanja odgovora zanemaruje se postupak kojim je odgovor dobijen, pa je način prevashodno primeren pitanjima iz teorije, koja zahtevaju jednostavnu reprodukciju znanja. Pri tom se, kod izbora ponuđenih odgovora, testiranom licu dozvoljava da nagađa odgovor. Takav način davanja odgovora nije primeren zadacima koji su česti u oblasti algoritama i struktura podataka, a kod kojih poznavanje i ispravna primena nekog algoritma predstavlja sastavni deo odgovora. Na primer, testirano lice koje uopšte ne poznaje dati algoritam bi bilo ocenjeno na isti način kao i lice koje bi pogrešilo u poslednjem koraku algoritma.

U ovom radu se predlaže arhitektura sistema za samostalnu obuku i proveru znanja studenata iz oblasti u kojima je od značaja postupak rešavanja zadatog problema. Iako je sistem prvenstveno razvijan kao softverski alat za unapređenje nastave na predmetu Algoritmi i strukture podataka na Elektrotehničkom fakultetu u Beogradu, predložena arhitektura omogućava širu primenu.

## II. Problemi i Postojeća Rešenja

Primena softverskih sistema za obuku i proveru znanja u obrazovanju ima veći broj poželjnih posledica:

- obuka se sprovodi bez ili uz minimalno učešće instruktora, koji svoje vreme može bolje da preraspodeli na aktivnosti koje zahtevaju više pažnje,
- obuka može da se prilagodi nivou predznanja obučavanog lica, koje može da ponavlja postupak obuke onoliko puta koliko je potrebno da savlada lekciju,
- rezultati testiranja su poznati neposredno nakon završetka testa, nezavisno od broja testiranih lica,
- obuka može da se sprovodi na daljinu, što efektivno smanjuje potrebu fizičkog prisustva i povećava vremensku dostupnost materijala za učenje.

Postojeći sistemi za proveru znanja, čiji su najznačajniji predstavnici dati u [7], tradicionalno sadrže skup unapred pripremljenih pitanja na koja lice, čije se znanje proverava, bira odgovor iz liste ponuđenih odgovora ili odgovor unosi u vidu kraćeg teksta. Ponekad je u pitanjima prisutna parametrizacija koja, umesto obične reprodukcije znanja, omogućava proveru razumevanja gradiva. Primer opisane parametrizacije je dat u [8]. Međutim, takav način provere znanja izaziva nekoliko značajnih problema. Najpre, nepodesan je za ocenjivanje zadataka u kojima se traži poznavanje nekog algoritma jer, u slučaju netačnog odgovora, ne uzima u obzir stepen poznavanja algoritma koji je testirano lice pokazalo, dok u slučaju tačnog odgovora nije poznato da li se do njega došlo slučajno (nizom grešaka) ili nagađanjem. Zatim, ograničena parametrizacija podrazumeva da instruktor mora da sastavi veći broj sličnih pitanja na istu temu. Takođe, prilikom zadavanja pitanja, instruktor mora da zada i tačan odgovor, što drastično smanjuje mogućnost samotestiranja (kada testirano lice samo sebi postavlja zadatak koji treba da reši), odnosno ograničava postupak obuke (obučavano lice ima na raspolaganju samo onaj materijal koji je pripremio

Miloš M. Milivojević je student master studija na Odseku za Računarsku tehniku i informatiku Elektrotehničkog fakulteta u Beogradu (e-mail: milivojevic.milos@gmail.com)

Đorđe M. Đurđević, Milo V. Tomašević, Elektrotehnički fakultet u Beogradu, Bulevar kralja Aleksandra 73, 11120 Beograd, Srbija; (telefon: 381-11-3218-385; e-mail: zorz@etf.rs, mvt@etf.rs)

instruktor). Motivacija za projektovanje arhitekture sistema za obuku i proveru znanja potiče iz činjenice da postojeći sistemi ne nude adekvatno rešenje za uočene probleme.

Veoma je poželjno da edukacioni sistem omogući samostalnu interaktivnu *pasivnu*, *upravljanu* ili *slobodnu* obuku. Pod pasivnom obukom se podrazumeva da obučavano lice posmatra automatizovan postupak izvršenja algoritma po koracima, uz mogućnost kontrole brzine izvršavanja i povratka na određeni korak. Pod upravljanom obukom se podrazumeva da obučavano lice određuje korake izvršenja, a pritom mu se postavljaju potpitanja koja ga vode ka odgovoru. Slobodna obuka se sprovodi bez dodatnih pitanja. Takođe, izuzev u slučaju pasivne obuke, obuka podrazumeva da se korisniku izdaje upozorenje čim napravi grešku, dok se kod provere znanja ne pružaju nikakve dodatne informacije.

U nastavku je dat kratak pregled pomenutih sistema koji podržavaju QTI [9] kao standard za predstavljanje testova i pitanja. Kao što je navedeno u [10], QTI definiše model podataka koji se koristi za predstavljanje pitanja, odgovora, ocenjivanja, rezultata i agregacije jedinica za procenu da bi se formirali testovi. Specifikacija je konstruisana tako da bude sposobna da podrži kako jednostavna, tako i složena pitanja i materijale za testove. Zahvaljujući implementaciji u vidu XML šeme, omogućeno je deljenje podataka između različitih sistema za procenu.

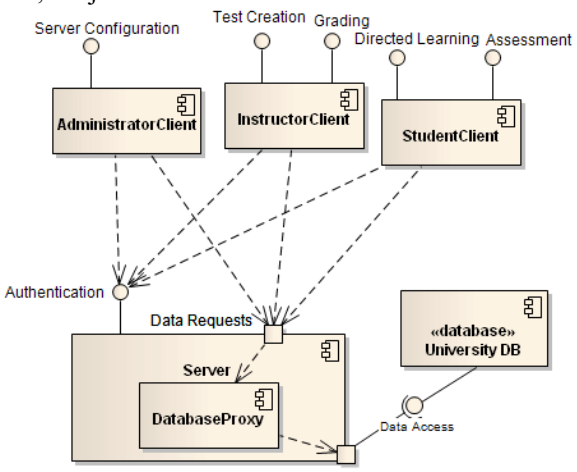
*Moodle* (Modular Object-Oriented Dynamic Learning Environment) [11], predstavlja besplatnu, *open-source* veb aplikaciju za elektronsko učenje, pisanu u PHP-u. Od januara 2010. godine, Moodle-ova korisnička baza sastoji se od 45,721 verifikovanih sajtova, koji opslužuju 32 miliona korisnika u 3 miliona kurseva. Koriste ga uglavnom obrazovne ustanove. *Dokeos* [12] je *open-source* veb aplikacija za elektronsko učenje, takođe razvijena u PHP-u. Trenutno opslužuje preko 2 miliona korisnika širom sveta. Koriste ga obrazovne ustanove, multinacionalne kompanije i federalne administracije. *Sakai Project* [13] predstavlja besplatnu, *community source* aplikaciju pisanu u Javi namenjenu nastavi, istraživanju i kolaboraciji. Nastao je kroz saradnju nekoliko američkih univerziteta. Trenutno ga koristi preko 160 obrazovnih institucija sa oko 200,000 korisnika. *OLAT* [14], je besplatna, *open-source* veb aplikacija pisana u Javi. Trenutno postoji 150 verifikovanih instalacija, sa preko 42,000 korisnika u 2000 kurseva.

Ni jedan od navedenih sistema ne nudi sve funkcionalnosti koje bi željeni sistem trebalo da ima, kao što su upravljana ili slobodna obuka i provera znanja iz oblasti u kojoj je od značaja postupak rešavanja problema. Autorima nije poznato da postoji javno dostupan predlog arhitekture takvog sistema. To je predstavljalo motiv za predlog idejnog rešenja softverskog sistema za samostalnu interaktivnu pasivnu, upravljanu ili slobodnu obuku i proveru znanja korisnika iz oblasti u kojima se vrednuje postupak dobijanja odgovora na postavljeno pitanje, konfigurabilnim metodom. Pritom, redosled aktivnosti koje čine postupak ne mora biti jednoznačan.

## III. Opis i Arhitektura Sistema

U osnovi, predloženi sistem omogućava prenošenje dela praktične nastave na interaktivan rad za računarom, gde simulator, koji koriste studenti, postaje lični instruktor. Obaveza nastavnika je da studentima obezbedi skup problema nad kojima oni treba da uče i provere svoje znanje, i test koji služi za ocenu stečenog znanja. Rad studenata se beleži i može da pruži nastavniku uvid u segmente gradiva koje studenti teže usvajaju, što je naročito bitno u režimu učenja. U režimu provere znanja, pristup testu je omogućen samo onim studentima koji pripadaju grupi korisnika kojima je taj test namenjen. Bodovanje studentskog rada se vrši poređenjem datog odgovora sa očekivanim odgovorom. Na osnovu ostvarenih bodova, ocenu daje sotfverska komponenta za ocenjivanje koju bira nastavnik.

Sistem se sastoji iz serverske i klijentske strane. Serverska strana podrazumeva jednu aplikaciju (serversku), dok se klijentska strana sastoji iz tri nezavisne aplikacije: studentske, nastavničke i administratorske. Razlog za uvođenje tri nezavisne klijentske aplikacije nalazi se u smanjenju složenosti implementacije, kao i povećanju sigurnosti sistema. S obzirom na tehnologiju izabranu za razvoj (videti poglavlje IV), reverzni inženjering klijentske aplikacije mogao bi otkriti određene detalje implementacije koji bi dozvolili varanje. Sa tri odvojene aplikacije, od kojih bi samo studentski klijent bio javno dostupan, potencijalno malicioznim korisnicima bio bi dostupan samo deo ukupnog klijentskog sistema, te reverzni inženjering ne bi narušio bezbednost sistema. Komponentni dijagram, koji prikazuje glavne delove sistema, dat je na slici 1.




Slika 1. Dijagram komponenti glavnih delova sistema.

Serverska aplikacija ima ulogu da autentifikuje i autorizuje klijente, kao i da obrađuje klijentske zahteve. Ona takođe može da, u slučaju duže neaktivnosti klijenta, prekine klijentsku sesiju uz prethodno izdavanje adekvatnog upozorenja. Radi povećane sigurnosti i fleksibilnosti, uvedena je posebna komponenta zadužena za obrađivanje zahteva za podacima – *DatabaseProxy*. U bazi podataka nalazi se repozitorijum pitanja, lista aktivnih

testova, podaci o studentskim grupama, podaci o korisnicima, statistički podaci, kao i privremeni podaci polaganja testova koji su u toku.

Administratorska aplikacija zadužena je za konfigurisanje serverske aplikacije, kao i za organizaciju korisnika (na primer, dodavanje ili brisanje nastavnika) i korisničkih grupa (na primer, dodavanje korisnika u grupu namenjenu potencijalnim "gost" korisnicima, tj. korisnicima koji nisu studenti, ali kojima je potrebno obezbediti pristup funkcionalnostima vidljivih studentima).

Nastavnička aplikacija zadužena je za stvaranje i održavanje studentskih grupa (prvenstveno vezanih za kurseve), stvaranje pitanja, sastavljanje i objavljivanje testova, kao i za pregled statistika i ocenjvanje rezultata testova. Pitanje se definiše zadavanjem algoritma i ulaznog skupa podataka nad kojim će se algoritam izvršavati, kao i oblasti kojoj pripada. Treba napomenuti da sistem podržava zadavanje pitanja na klasičan način (na primer, u vidu teksta sa ponuđenim odgovorima), ali to nije predmet ovog rada. Test se sastoji od proizvoljnog broja pitanja. Prilikom dodavanja pitanja u test, nastavniku je omogućeno da iz repozitorijuma pitanja izabere postojeće pitanje ili da napravi novo pitanje. Prilikom objavljivanja testa, nastavnik određuje studentsku grupu na koju se test odnosi, dužinu trajanja testa, datum njegovog polaganja ili vremenski interval njegove dostupnosti, kao i tip testa (za ocenu ili samostalno vežbanje). Nastavnik definiše način ocenjivanja studentskih radova izborom jednog od ponuđenih konfigurabilnih modula za ocenjivanje.

Studentska aplikacija namenjena je obuci i testiranju i obezbeđuje interaktivan rad (prikaz i pokretanje testova) putem grafičkog interfejsa za koji je kao osnova upotrebljena aplikacija razvijena u [3], [5]. Nakon autentifikacije, studentska aplikacija od servera preuzima listu dostupnih testova, od kojih student bira jedan za rešavanje. Studentska aplikacija automatski preuzima sve podatke potrebne za pokretanje testa, uključujući i implementacije algoritama. Treba napomenuti da aplikacija može da radi samostalno, bez povezivanja sa serverskom aplikacijom. Aplikacija je predviđena za dva režima rada: obuku i testiranje. Kao što je prethodno napomenuto, obuka može biti pasivna, upravljana ili slobodna. Testiranje može biti samostalno ili za ocenu. Samostalno testiranje se razlikuje od testiranja za ocenu u tome što vreme polaganja testa nije ograničeno. U ovom režimu student rešava test pitanje po pitanje, bez pomoći, a potom, na kraju testa, dobija rezultate rešavanja u vidu procenta poklapanja svojih akcija sa traženim i brojem napravljenih grešaka, grupisanih po njihovoj ozbiljnosti.

Za određivanje procenta poklapanja korisničkih akcija i traženih akcija zadužena je implementacija algoritma koji je zadat prilikom pravljenja pitanja, dok su za procenu ozbiljnosti greške zadužene implementacije struktura podataka nad kojima se algoritam izvršava. Rezultati rešavanja nekog testa čuvaju se u bazi podataka, odakle su dostupne nastavnicima odgovornim za njihovo ocenjivanje.

## IV. Detalji Implementacije

Povezivanje klijentskih aplikacija sa serverskom se zasniva na biblioteci programskog jezika Java, namenjenoj za udaljeni poziv metoda (*remote method invocation*, skraćeno RMI). Udaljeni poziv metoda ima tu pogodnu osobinu da omogućava da se komunikacija dvaju klasa koje se nalaze i čije metode se izvršavaju na različitim računarima obavlja na isti način kao da su te klase na jednom računaru. Za razvoj je korišćeno razvojno okruženje NetBeans.6.8, i Java SE2 1.6.

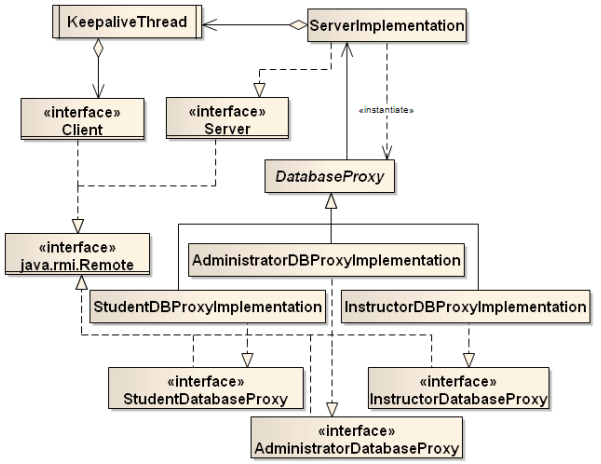
Na slici 2 je prikazan uprošćeni UML dijagram ključnih klasa serverske aplikacije. Interfejsi *Server* i *Client* propisuju komunikacioni API koji se mora poštovati u realizaciji serverskih i klijentskih klasa. Klasa *ServerImplementation* implementira serverski interfejs. Njeno zaduženje je da autentifikuje i autorizuje klijentsku aplikaciju i obezbedi joj pristup bazi podataka.

Pristup bazi podataka se vrši posredstvom klase *DatabaseProxy*, odnosno klasa koje je nasleđuju (*AdministratorDBProxyImplementation*, *InstructorDBProxyImplementation* i *StudentDBProxyImplementation*), a koje instancira serverska aplikacija nakon uspešne autentifikacije, prema tipu klijenta. Svaka od ovih klasa implementira odgovarajući interfejs – *AdministratorDatabaseProxy*, *InstructorDatabaseProxy* i *StudentDatabaseProxy* – koji propisuje API za udaljenu komunikaciju sa klijentskim aplikacijama. Razlog ovakve hijerarhije je povećana bezbednost aplikacije. Svaka od klijentskih aplikacija preko svoje DBProxy implementacije ima pristup tačno određenom skupu funkcionalnosti. Tako *StudentDatabaseProxy* interfejs propisuje funkcionalnosti potrebne studentu – dohvatanje željenog testa iz baze, upisivanje rezultata polaganja testa u bazu i sl., dok *InstructorDatabaseProxy* propisuje funkcionalnosti potrebne nastavniku – dodavanje pitanja u repozitorijum, objava testova, upis ocena i sl.

Uloga aktivne *KeepaliveThread* klase je da vodi računa o trajanju perioda neaktivnosti klijentskih aplikacija i da, po potrebi, prekida klijentsku sesiju.

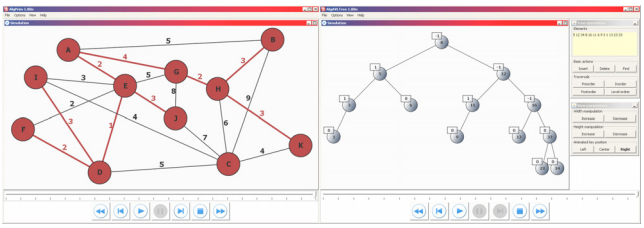
Sistem prikazan u ovom radu oslanja se na alat *Vizuelni Simulator Algoritama* (VSA). Ovaj alat je rezultat niza diplomskih radova i završnih radova osnovnih akademskih studija studenata Elektrotehničkog fakulteta u Beogradu [2]-[6].

Jezgro VSA sastoji se iz tri ključne osnovne klase i njihovih specijalizacija – klase *Algorithm*, koja propisuje sam algoritam, tj. postupak izvršavanja, klase *Observable-Variable*, koja je predstava strukture podataka koju algoritam upotrebljava i klase *Action*, koja je apstrakcija akcije algoritma nad određenom strukturom podataka. Pokretanje algoritma nad zadatom strukturom podataka rezultovaće listom akcija koje se naknadno tumače od strane prezentacionog sloja u cilju formiranja adekvatnog prikaza.




Slika 2. Uprošćeni dijagram ključnih klasa serverske aplikacije

Ključni deo prezentacionog sloja predstavljen je klasom pogleda *AbstractView* i njenim specijalizacijama. Klasa pogleda zadužena je za prikaz liste akcija formirane kao rezultat izvršenja nekog algoritma. Ova klasa iterira listom akcija i prikazuje stanje korišćenih struktura posle svakog koraka algoritma, uz potencijalno animiranje akcije koja je do tog stanja dovela. Na slici 3 su prikazani primeri primene VSA pri vizualizaciji algoritama nad grafovima i binarnim stablima.



*a)* *b)*

Slika 3. Primeri primene VSA za prikaz izvršenja nekih algoritama; a) pronalaženje minimalnog obuhvatnog stabla u zadatom grafu; b) umetanje ključeva u AVL stablo.

Opisana hijerarhija klasa VSA igra ključnu ulogu i u sistemu koji je tema ovog rada. Provera valjanosti studentovog odgovora zasniva se na proširenju klasa *Algorithm* i *ObservableVariable*. Klasa *Algorithm* je zadužena za proveru procenta poklapanja niza studentovih akcija sa nizom validnih akcija, dok je *ObsevableVariable* zadužena za procenu ozbiljnosti greške, koja se delimično zasniva na proveri valjanosti rezultujuće strukture podataka (na primer, ukoliko je niz studentovih akcija doveo do loše formirane strukture, to je znak ozbiljne greške). Prezentacioni sloj VSA je korišćen kao osnova prezentacionog sloja sistema predloženog u ovom radu.

## V. Zaključak

U radu je prikazana arhitektura softverskog sistema za obuku i proveru znanja iz oblasti u kojima je od značaja postupak rešavanja postavljenog problema, što nije slučaj sa tradicionalnim sistemima. Predloženi sistem omogućava slobodnu i upravljanu obuku kod koje obučavano lice ima mogućnost da zada parametre i podatke nad kojima se algoritam izvršava. Kod testiranja, sistem može da oceni postupke koji su delimično saglasni sa datim algoritmom i proceni njihovu uspešnost, što pruža bolji uvid u pokazano znanje i razumevanje gradiva od strane obučavanog lica.

Unapređenje sistema bi trebalo da obuhvati razvoj sledećih komponenti: utvrđivanje performansi algoritama (uz mogućnost poređenja performansi srodnih algoritama), statistička analiza rezultata studentskih testova, praćenje rada i napredovanja svakog studenta pojedinačno. U cilju postizanja kompletnosti sistema, potrebno je implementirati sve relevantne algoritme i formirati odgovarajuću bazu testova.

## Literatura


[1] Dillenbourg, P., Schneider, D.K., Synteta, P. “Virtual Learning Environments,” in Proc. 3rd Hellenic Conference "Information & Communication Technologies in Education", pp. 3-18, 2002.

[2] Mićanović, I., “Projektovanje i implementacija jezgra za VSA,” diplomski rad, Elektrotehnički fakultet u Beogradu, 2009.

[3] Bjegović, M., “Razvoj grafičkog interfejsa i prezentacionog sloja VSA,” diplomski rad, Elektrotehnički fakultet u Beogradu, 2009.

[4] Đurišić, D., “Projektovanje i implementacija unapređenog jezgra VSA,” završni rad osnovnih akademskih studija, Elektrotehnički fakultet u Beogradu, 2009.

[5] Milivojević, M., “Projektovanje i implementacija unapređenog grafičkog interfejsa VSA,” završni rad osnovnih akademskih studija, Elektrotehnički fakultet u Beogradu, 2009.

[6] Mirosavljević, D., “Implementacija algoritama zasnovanih na stablu u okviru VSA,” završni rad osnovnih akademskih studija, Elektrotehnički fakultet u Beogradu, 2009.

[7] http://en.wikipedia.org/wiki/Virtual_learning_environment#List_of_some_virtual_learning_environments

[8] Moodle Question Type – Java Molecule Editor, http://moodle.org/mod/data/view.php?d=13&rid=296

[9] IMS QTI, IMS Global Learning Consortium Question & Test Interoperability Specification (QTI), 2005, http://www.imsglobal.org/question/.

[10] Davies, W. M., Davis, H. C., “Designing Assessment Tools in a Service Oriented Architecture, ” In Proc. 1st International ELeGI Conference on Advanced Technology for Enhanced Learning Napoli, Italy, 2005.

[11] Moodle.org: open-source community-based tools for learning, http://moodle.org/

[12] dokeos | Open Source E-Learning, http://www.dokeos.com/.

[13] Sakai Project - an Open Source suite of learning, portfolio, library and project tools, http://sakaiproject.org/

[14] OLAT - Your Open Source LMS, http://www.olat.org/


## Abstract


In this paper we propose a new software system for self-directed interactive learning and assessment in domain of algorithms and data structures. The system extends *Vizuelni Simulator Algoritama* (VSA), a tool developed at the School of Electrical Engineering, University of Belgrade. The system provides control of trainee’s independence level, flexible input data set assignment and configurable automatic assessment of testee’s actions. The paper describes the architecture of the proposed system and gives relevant implementation details.


## A System for Interactive Training and Testing in Algorithms and Data Structures


Miloš Milivojević, Đorđe Đurđević, Milo Tomašević