Samodejno preverjanje znanja pri pouku programiranja

Andrej Brodnik, Univerza v Ljubljani, FRI in Univerza na Primorskem PINT
andrej.brodnik@upr.si
Bojan Sinkovič, Univerza na Primorskem, PEF
bojan.sinkovic@student.upr.si
Jernej Vičič, Univerza na Primorskem, FAMNIT
jernej.vicic@upr.si

Povzetek

Članek predstavlja poskus uvedbe samodejnega sodniškega sistema v proces poučevanja programiranja. Pri tem se osredotočamo na dve fazi poučevanja: samo poučevanje in izvedbo tekmovanja iz programiranja. Izdelano ter predstavljeno je okolje izgrajeno v okviru e-izobraževalno Moodle, ki vključuje sodniški sistem za samodejno preverjanje programerskih nalog. Slednji je transparentno umeščen v e-izobraževalno okolje Moodle. Okolje je bilo delno preizkušeno na letošnjem državnem ACM tekmovanju iz računalništva za srednješolce in na dveh generacijah študentov. Izvedena je bila tudi krajša anketa. V članku je predstavljeno tekmovalno okolje in objektivni rezultati ankete ter subjektivna mnenja obeh vrst uporabnikov, študentov – reševalcev nalog, asistentov ter profesorjev – snovalcev nalog in učnega procesa.

Ključne besede: *Tekmovanje iz računalništva, učenje programiranja, samodejna evalvacija, samoevalvacija, ACM.*

1 Uvod

Večina avtorjev relevantne literature, kot na primer (Booth 2009), se strinja, da se učenje programiranja razlikuje od klasičnih oblik učenja kot na primer matematičnih vsebin saj zahteva veliko več individualnega dela, tj. samega programiranja. Takšen način poučevanja zahteva uvedbo daljših in krajših projektnih nalog ter njihov nadzor. V ta namen smo izdelali ter preizkusili sistem za samodejno razdeljevanje ter ocenjevanje programerskih nalog. Sodniški sistem je transparentno umeščen v e-izobraževalno okolje Moodle kar omogoča enostaven nadzor na sodelujočimi ter poznan nadzor nad rezultati (rezultati so združeni v Moodlov pregled ocen). Posebna pozornost je bila namenjena nadzoru nad plagiatorstvom.

Članek je razdeljen na sledeč način: drugi razdelek, Opis področja, uvede bralca v predstavljeno raziskovalno področje, predstavljeno je učenje programiranja, programerska tekmovanja ter posebej predstavi problem plagiatorstva pri pisanju domačih nalog. Sledi opis osnovnih razlogov za postavitev takšnega sistema ter izvedbo opisanega poskusa. Četrti razdelek predstavlja izdelan sistem, sledi Metodologija evalvacije in rezultati. Članek se zaključi z diskusijo ter predstavitvijo nadaljnjega dela.

2 Opis področja ter motivacija

2.1 Učenje programiranja

Učenje in poučevanje programiranja sta zahtevna procesa. Na obeh straneh (poučevalčevi in študentovi) terjata mnogo časa za priprave, potrebnega je veliko potrpljenja in predvsem vztrajnosti. Vsi našteti pogoji odvrnejo marsikaterega študenta take usmeritve od dokončanja študija. Preostali del znanja željne populacije pa si prizadeva pridobiti zahtevano znanje. Posredovalci znanja jim zato hočejo predati vse potrebno, da bodo osvojili predavane pojme, razvili ustrezno mišljenje in bili pripravljeni na izpolnjevanje zahtev pridobljenega naziva.

Kljub vsemu temu pa se predano znanje ne prenese na čisto vse poslušalce. Nasprotno: število študentov, ki resnično osvoji zahtevano znanje, je zelo majhno. Po (Kaasboll 1999) so to lahko simptomi preskopega raziskovanja na področju poučevanja programiranja. Slednje nima močne tradicije, kot jo ima npr. matematika.

Informacijske tehnologije so zelo mlada veja znanosti. Novosti se pojavljajo zelo pogosto, kar pomeni prilagajanje učnih vsebin trenutnim trendom. Primer tega je bilo npr. uvajanje objektno naravnanega programiranja v uvodne predmete računalništva, sedaj je to npr. grafika in multimedia, kot je zapisano v (Pedroni 2003). Osvajanje takih konceptov je (za večino) zahtevno, zato morajo izvajalci predmetov biti izredno previdni pri planiranju učnih aktivnosti in upoštevati dosedanje rezultate analiz in študij.

2.2 Programerska tekmovanja

Obstaja veliko različnih tipov tekmovanj v programiranju: skupinska, individualna, nekajurna, večdnevna itd. Cormack (Cormack 2006) razlikuje tekmovanja ne le po izvedbenih lastnostih, ampak tudi po ciljih in namenih. Bodisi je to druženje vrstnikov oz. somišljenikov, bodisi je v ozadju finančna motivacija, vsako tekmovanje prinese veliko dobrih izkušenj (sklepanje prijateljstev, spoznavanje potencialnih sodelavcev, pridobivanje materialnih dobrin, plačana potovanja po Evropi in ob primerni uvrstitvi tudi dlje). V nadaljevanju omenjamo tri tekmovanja svetovne ravni – povzeto po (Cormack 2006) in spletnih straneh samih tekmovanj.

Korenine tekmovanja *ACM ICPC* (*The ACM-ICPC International Collegiate Programming Contest*) segajo v leto 1970. Priredili so ga pri delu Alpha Chapter društva UPE Computer Science Honor Society. Ideja je dosegla visoko zanimanje, saj je predstavljala inovativno iniciativo primerjanja najboljših študentov na vzhajajočem področju računalništva. Kasneje se je tekmovanje pod okriljem društva ACM (s sedežem na Baylordski univerzi od leta 1989) razširilo na globalno mrežo univerz. Na njih so gostili regionalna tekmovanja, ki so služila določanju najboljših ekip in te promovirala na svetovni finale (*ACM-ICPC World Finals*). Odkar so se leta 1997 pri podjetju IBM odločili sponzorirati tekmovanje, se je udeležitev zelo povečala - nazadnje je sodelovalo 1.838 univerz iz 88 držav (http://cm2prod.baylor.edu/ICPCWiki/attach/staticResources/Factsheet.pdf).

Kot ACM ICPC predstavlja svetovno tekmovanje na univerzitetni ravni, predstavlja tekmovanje *IOI* (*International Olympiad in Informatics*) eno izmed petih znanstvenih olimpijad za srednješolce. Obstajajo še *International Mathematical Olympiad*, *International Physics Olympiad*, *International Chemistry Olympiad* in *International Biology Olympiad.* IOI je med njimi najmlajša s svojim začetkom v letu 1989.[1] Predlog za njeno ustanovitev so podali pri organizaciji UNESCO (*United Nations Educational, Scientific and Cultural Organization*). Ciljna skupina so srednješolski učenci iz celega sveta. Princip tekmovanja se drži osnovnih lastnosti, a ima nekaj posebnosti. Za razliko od ACM ICPC tekmovalci dobijo točke tudi pri delno pravilnih rešitvah. Večina vseh nalog je sojenih po izhodnih podatkih, kjer se točkujejo le pravilni deli in ne v celoti.

TopCoder je spletna skupnost, kjer se člani lahko pomerijo v vsakotedenskih tekmovanjih za naraščanje svojega števila točk. Te kasneje pridejo v poštev pri večjih tekmovanjih organiziranih s strani lastnikov portala ali na pobudo kakega podjetja. Takih tekmovanj se lahko udeležijo člani z dovolj visokim številom prej doseženih točk.

2.3 Slovenija

V Sloveniji je učenje programiranja zelo izrazito osredotočeno na univerzitetni nivo. Obstajajo sicer vsebine programiranja tudi v srednješolskem in osnovnošolskem kurikulu (Tomazin et al. 2007), vendar je ta zastopanost veliko slabša od mednarodno sprejetih standardov (ACM K12 kurikulum, http://ucilnica.fri.uni-lj.si/mod/resource/view.php?id=5946). Kljub temu v Sloveniji že več kot 20 let potekajo tekmovanja iz znanja računalništva za srednješolce (prim. opombo 1) in letos je prvič slednje prešlo pod strokovno pokroviteljstvo ACM (http://rtk.ijs.si/). Tekmovanje bi moralo veljati za vrhunec kurikuluma, vendar žal zaradi neprisotnosti računalniških znanosti v osnovnošolskem in srednješolskem kurikulumu temu ni tako (http://ucilnica.fri.uni-lj.si/mod/resource/view.php?id=5944).

Pod pokroviteljstvom ACM imamo v Sloveniji tudi nekajletno tradicijo univerzitetnih tekmovanj iz programiranja. SUPP (Slovensko Univerzitetno Prvenstvo v Programiranju) in kasneje UPM (Univerzitetni Programerski Maraton). Tekmovanje predstavlja nacionalni nivo mednarodnega ACM programerskega tekmovanja (*The ACM-ICPC International Collegiate Programming Contest*, http://cm2prod.baylor.edu).

2.4 Osnovni princip tekmovanj

Pri vseh tekmovanjih velja princip, da so tekmovalci razdeljeni v skupine in skušajo v nekem časovnem obdobju v čim krajšem času rešiti čim več nalog. Preverjanje pravilnosti rešitve je lahko samodejno ali ročno. Morda se zdi ročno preverjanje pravilnosti ostanek zgodovine, vendar je iz didaktičnega zornega kota zelo smiselno. Še posebej pri začetnikih, ki imajo nemalo težav ne samo pri reševanju same naloge, ampak tudi pri postopku kodiranja in obvladovanja prevajalno-izvajalnega sistema.

V primeru, da preverjamo pravilnost rešitev samodejno, moramo poudariti, da lahko ne glede na tip tekmovanja potegnemo črto in izpostavimo skupne imenovalce, ki so del vseh tekmovanj[2]. To so: i) *sodniški sistem* (okolje za izvajanje in evalvacijo rešitev, točkovanje); ii) *skupine* (lahko so sestavljene iz enega samega člana); iii) *množica problemov*; iv) *množica rešitev;* v) *testni primeri* (javni in tajni); ter vi) tekmovalčevo okolje. Pri tem bi dodali še komunikacijo med skupinami, ki ni nujna vendar je zelo zaželena (običajno v obliki foruma).

2.4.1 Sodniški sistem

V osnovi je to program namenjen izvajanju programov v izoliranem okolju s širokim naborom testnih podatkov. Ponavadi je postavljen na operacijskem sistemu UNIX. Razlogi za to so večji nadzor nad izvajalnim okoljem, dostopnost prevajalnikov za različne programske jezike ter cena operacijskega sistema.

Običajno zna sistem glede na rezultate pri obdelavi oddanih programov vrniti pet odgovorov: napaka pri prevajanju (*compile error*): program se ni prevedel uspešno; napaka pred/med izvajanjem (*runtime error*): program ni začel z izvajanjem oz. se je sesul med izvajanjem; napačna rešitev (*wrong answer*): program ni vrnil pričakovanih izhodnih podatkov; predolg čas izvajanja (*time limit exceeded*): čas izvajanja programa je presegel zgornjo mejo predvideno za pridobitev izhodnih podatkov; pravilna rešitev (*program accepted*): program je vrnil pričakovane izhodne podatke v predvidenem času. Naloga je uspešno rešena le v primeru odgovora *pravilna rešitev*. Pri vseh ostalih odgovorih dodeli sodniški sistem skupini kazenske točke.

2.5 Plagiatorstvo

Plagiatorstvo oz. *plagiatizem*, kot ga poimenujejo v (Munda in Lenič, 2005), je razglašanje idej in rešitev drugih za lastne. Ne zajema le izdelkov v fizični obliki (npr. ure znanih znamk s preoblikovanim imenom znamke) temveč tudi medije (npr. članke in knjige) in elektronske vsebine (npr. izvorne kode programov).

Dodatna funkcionalnost, ki si jo torej želimo je preverjanje o obstoju plagiatov. Plagiatorstvo je prisotno na vseh področjih človeškega ustvarjanja in programiranje ni izjema. Prilaščanje tujih stvaritev pri poučevanju programiranja povzroča škodo pri obeh udeležencih: avtor je oškodovan pri lastnini ideje ter možnostjo obtožbe prilaščanja (najprej je treba ugotoviti kdo je originalno idejo izpeljal), plagiator pa se tako izogne učenju, vadbi in premišljevanju. Preprečevanje tega početja je nujen proces pri samodejnem preverjanju oddanih rešitev.

3 Opis sistema

Spletno učno okolje Moodle omogoča razširitve ponujenih storitev in vsebin. Poleg obstoječih modulov je na voljo še kopica drugih (uradnih in neuradnih), ki so dostopni na uradnem CVS repozitoriju (http://cvs.moodle.org/). Zastavljeni cilji modula za samodejno preverjanje oddanih nalog so skupni večini lastnosti tekmovanj iz programiranja: *predstavitev problema/naloge, javni testni podatki, tajni testni podatki (v sklopu), časovna omejitev izvajanja programov, točkovni sistem, različna sporočila s strani sodniškega sistema in testiranje plagiatorstva*. Slednje je dodatna zahteva, ki omogoča dokaj zanesljivo odkrivanje „prepisovanja“ in je uporabna ne toliko na tekmovanjih, kot pri pouku programiranja.

3.1 Opis

Po pregledu modulov na razpolago je za implementacijo sistema za samodejno preverjanje oddanih nalog bil najbolj primeren Epaile opisan v (Garro 2007), ki ga je spisal Arkaitz Garro; z njim je sodeloval na natečaju Google Summer of Code.

Epaile je razširitveni modul spisan za Moodle 1.9. Izpeljan je iz modula Dejavnost (*Activity*), kot podtip nosi ime Program. Za evalvacijo oddanih rešitev skrbi znani odprtokodni sodniški sistem DOMjudge, ki so ga uporabili/ga uporabljajo na severozahodnem delu Evropskega tekmovanja iz programiranja ACM ICPC v letih 2007, 2008 in 2009.

Kljub ustreznosti ima modul kar nekaj pomanjkljivosti glede na naše zahteve:

1. Modul Program sprejema tajne vhodne podatke v obliki posameznih testnih primerov. To bi sicer lahko enostavno prikazalo pri katerem testnem primeru je prišlo do problema, vendar ne omogoča njihovega združevanja v en sam večji sklop podatkov, kot je v uporabi pri vseh (bolj znanih) tekmovanjih iz programiranja.

2. Izvršitev oddanega programa poteka sinhrono z oddajo. Ko uporabnik odda datoteko se sproži proces, ki pokliče evalvacijsko skripto sodniškega sistema in tako zakasni odgovor strežnika do konca izvajanja skripte. To pomeni, da si morata sodniški sistem in Moodle deliti isti strežnik, kar lahko povzroči zastoj odzivanja strežnika tudi za uporabnike, ki z oddajami programov nimajo nikakršne povezave.

3. Vsak Program lahko vsebuje le eno zadolžitev, kar onemogoča pripravo njihovih sklopov, reciklažo posameznih nalog itd.

4. Nabor preverljivih programskih jezikov je sicer pogojen s sodniškim sistemom, vendar je kljub njihovi številčnosti Epaile dovoljuje izbiro le enega izmed teh, tj. vsak modul Program omogoča preverjanje rešitve izdelane le v enem programskem jeziku.

Z upoštevanjem pomanjkljivosti in uporabo obstoječega modula Epaile smo razvili njegovo nadgradnjo in se tako približali implementaciji modula s samodejnim preverjanjem oddanih nalog po vzoru tekmovanj iz programiranja.

3.1.1 Zadolžitve

Zaradi razširitve modula je bilo potrebno pregledati ustreznost drugih modulov v sistemu Moodle. Modul Vprašanje (*Question*) je deloval posebej uporaben. Moč jih je združevati v Kategorije (*Category*). Moduli Vprašanje so bili sprva mišljeni za interno uporabo modulov Kviz (*Quiz*), vendar so se razvijalci zaradi splošnosti in modularnosti odločili, da jih bodo preprogramirali v samostojne enote pripravljene za uporabo v drugih modulih. Na žalost proces preprogramiranja še zdaleč ni končan in je možno uporabiti modul Vprašanje le v modulu Lekcija (*Lesson*).

Po zgledu modulov Vprašanje je bila implementirana poenostavljena različica, spisana posebej za modul Program in nosi ime Zadolžitev (*Task*). Kreiranje nove zadolžitve zahteva od upravljavca ime, opis, javne testne vhodne in izhodne podatke, tajne testne vhodne in izhodne podatke, nabor programskih jezikov, za katere je možno uspešno preveriti rešitev ter časovno omejitev izvajanja programa. Zaradi enostavnosti je vsako zadolžitev moč dodeliti kateremukoli modulu Program. Pred navadnim uporabnikom je skrit. Namesto tega ima za vsako zadolžitev predstavitveni del, kjer so navedeni ime, opis ter javni testni podatki.

3.1.2 Testiranje oddanih rešitev

Sodniški sistem *DOMjudge* opisan v (Eldering et al. 2004) že sam po sebi ponuja testno ogrodje za spletno ustvarjanje tekmovanj, njihovo administracijo, sodelovanje in spremljanje. Sistem je zasnovan modularno (jedrni del je ločen od vmesnika). Različica, ki jo uporablja Epaile je 2.1.0. Sodniški sitem je doživel nekaj posodobitev in je zato v novem modulu uporabljena najnovejša različica, tj. 2.2.3. Prevajanje in evalvacija programov poteka sedaj ločeno od oddajanja datotek. Izobraževalni ter sodniški sistem sta lahko na različnih strežnikih. Celo več, sodniški sistem je lahko računalniška gruča.

Vnos tajnih testnih podatkov v modulu Epaile ni bil uporabniku prijazen, saj je zahteval od upravljavca posamično podajanje testnih primerov (in ne celotnih testnih podatkov). Zato je v novem modulu moč oddati datoteki s testnimi podatki (vhodnimi in izhodnimi) na strežnik kar preko vnosne maske Zadolžitev.

3.1.3 Preverjanje plagiatov

V grobem lahko preverjamo plagiatorstvo ob oddaji rešitve ali kasneje. Takojšnje preverjanje ob oddaji pri večjem številu udeležencev lahko povzroči preobremenjenost nosilnega strežnika sistema Moodle. Ob tem obstaja možnost *takojšnjega opozarjanja*, ki bi uporabnika opozori o rešitvi, ki je zelo podobna njegovi. Uporabnik se nato loti izdelave lastne rešitve ali (v najslabšem primeru) poskusil poiskati novega „darovalca“.

Poleg tega mora obstajati še možnost ročnega poganjanja primerjav, ki ga sproži skrbnik modula. Ob tem mora snovalec dobiti kar se da mnogo informacij o vseh oddajah in njihovih primerjavah.

Univerza Stanford ponuja spletno storitev MOSS (*Measure Of Software Similarity*) opisano v (Bowyer et al. 2000) in (Aiken et al. 2006), ki deluje od leta 1997 kot storitev (v razvoju od leta 1994) za odkrivanje plagiatov na področju programskih zadolžitev. Za (osnovne) parametre sprejme programski jezik (C, C++, Java id.) ter datoteke, ki jih želimo primerjati. Storitev vrne kot rezultat naslov do spletne strani (v resnici jih je več), na kateri so nanizane primerjave oddanih datotek v obliki odstotkov podobnosti datotek. Poleg tega si lahko ogledamo mesta (posamezni kosi izvorne kode), kjer sta si datoteki najbolj podobni. Rezultati obdelave se hranijo na strežniku še 2 meseca po poizvedbi. V (Munda et al. 2007) je povzet princip delovanja primerjave izvorne kode oddanih datotek. Sloni na zaporedjih simbolov oz. t.i. k-gramih. K-grami so zaporedja simbolov dolžine k, iz katerih izračunamo enolično predstavitev, v kateri je tudi informacija o lokaciji zaporedja. Če si dva programa delita več takih izsekov, potem si najverjetneje delita tudi več k-gramov.

Implementacija preverjanja oddanih rešitev v sistemu Moodle je zahtevala stvaritev spletne storitve (*web service*) s klicem skripte MOSS, ki oddane rešitve združuje po programskih jezikih. Glede na to, da je primerjava datotek zajeta na spletni strani storitve MOSS, je kot odgovor v sistemu Moodle dovolj podati povezavo do primerjave. Klic spletne storitve vsak odgovor (spletni naslov) zapiše v bazo, kar zagotavlja hranjenje zgodovine primerjav.

3.2 Uporaba

Uporabniški vmesnik modula Program je zasnovan intuitivno in v skladu z ostalimi moduli. Kljub modernim tehnologijam (*AJAX*), naprednim vmesnikom in ogrodjem (*frameworks*) na razpolago dandanes je potrebno zagotoviti zlitost v že obstoječ izgled celotnega sistema Moodle; alternativa bi bila stvaritev novega sistema, npr. (Connolly, 2008). Vmesnik je razdeljen na dva dela: predstavitveni oz. vnosni (skupen vsem uporabnikom) ter urejevalni.

3.2.1 Predstavitveni, vnosni vmesnik

Vsak modul program ima nabor Zadolžitev. Na predstavitvenem delu je tabela, ki prikazuje nabor. V njej so podatki o času oddaje in sporočilom sodnika vseh dosedanjih oddaj. Dostop do posamezne Zadolžitve je omogočen preko povezave z njenim imenom.

Predstavitveni del je razdeljen na vnosni obrazec ter predstavitveno vsebino Zadolžitve. Vnosni obrazec sestoji iz treh bistvenih delov: izbirnega polja (kjer so nanizane vse Zadolžitve), polja za oddajo datoteke z izvorno kodo programa (prenos iz lokalnega računalnika na strežnik) ter gumbom za potrditev oddaje.

Pod vnosnim obrazcem imamo standarden prikaz Zadolžitve. Na vrhu je ime Zadolžitve, nato sledi njen opis, primer vhodnih testnih podatkov in primer izhodnih testnih podatkov.

3.2.2 Urejevalni vmesnik

Vmesnik za urejanje Zadolžitev je viden le učiteljem. Sestoji iz dveh delov: spiska zadolžitev modula Program ter nabora vseh Zadolžitev. Po zgledu modula Vprašanje je dodajanje, odvzemanje, brisanje, urejanje ter ustvarjanje implementirano na podoben način.

Ob ustvarjanju nove Zadolžitve mora snovalec programske naloge preko vnosnega obrazca podati njeno ime, opis, javne testne vhodne in izhodne podatke, tajne testne vhodne in izhodne podatke, nabor programskih jezikov, za katere je možno uspešno preveriti rešitev ter časovno omejitev izvajanja programa. Po potrditvi se dodana zadolžitev pojavi na spisku vseh zadolžitev in jo snovalec lahko doda v nabor Zadolžitev modula Program.

4 Uporaba sistema Moodle pri pouku programiranja in pri tekmovanju

V dosedanjem opisu sistema smo se omejili samo na opis modula za samodejno preverjanje pravilnosti oddane programske rešitve. Kot smo zapisali že na začetku je didaktično zelo potrebno pri pouku programiranja združevati samodejno preverjanje pravilnosti delovanja programa in običajno preverjanje pravilnosti in smiselnosti programsko-algoritmičnega razmišljanja. Za slednje v sistemu Moodle uporabimo običajno aktivnost oddaje naloge. Obe aktivnosti lahko uporabimo tako pri pouku programiranja, kot pri izvedbi računalniškega tekmovanja.

4.1 Tekmovanje

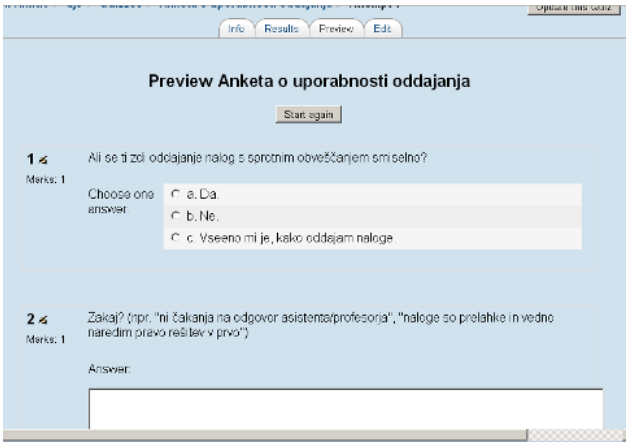
Na letošnjem 4. tekmovanju IJS iz znanja računalništva, ki je bilo izvedeno pod strokovnim pokroviteljstvom ACM Slovenija, smo prvič na celotnem tekmovanju uporabili računalniško podporo. Tekmovanje je razdeljeno na tri skupine, pri čemer se v prvih dveh skupinah preverja ne sâmo pravilnost in delovanje rešitve, ampak predvsem algoritmično razmišljanje tekmovalcev. Tekmovalno okolje je bilo razdeljeno na strežniški del in tekmovalčevo okolje. Strežniški del je predstavljala namestitev Moodle, v kateri so bili pripravljeni predmeti – za vsako tekmovalno skupino po eden. Zaradi primerljivosti je bilo tekmovalčevo okolje izvedeno kot navidezni stroj (*virtual machine*), v katerega smo namestili dovoljeno programsko opremo.

Čeravno je bil v prejšnjem razdelku opisani modul prvotno namenjen izvedbi tekmovanja, smo se letos odločili, da ga zaradi sorazmerno kratkega časa namenjenega za preverjanje delovanja sistema, ne bomo uporabili. Uporabili smo ločeno razvito in neintegrirano okolje, katero se uporablja že več let. Za prihodnje leto načrtujemo integracijo okolja v sistem Moodle.

Izkušnje s tekmovanja so bile zelo pozitivne. Na tekmovanju je sodelovalo 60 tekmovalcev v prvi skupini in 15 v drugi skupini. Imeli so možnost oddaje nalog v papirni obliki ali preko spletnega vmesnika in vsi razen enega so se odločili za slednje. Samo pregledovanje nalog se je izkazalo tudi za veliko lažje, saj je v prejšnjih letih bilo precej težav s prebiranjem rokopisa tekmovalcev. Bolj kot zanimivost naj omenimo verjetno edino slabo izkušnjo pri nalogi, ki je zahtevala znanje pisanja iterativne zanke. Eden od tekmovalcev se je rešitvi izognil tako, da je namesto iteracije samo večkrat prekopiral.

4.2 Poučevanje

Sistem je bil uporabljen pri dveh izvedbah predmeta Programiranje 1 (*P1*; osnove programiranja), eni izvedbi predmeta Programiranje 2 (*P2*; koncepti programskih jezikov) v prvem letniku ter pri predmetu Podatkovne strukture in algoritmi (*PSA*) v drugem letniku dodiplomskega bolonjskega programa Računalništvo in informatika na Univerzi na Primorskem, Fakulteti za matematiko, naravoslovje in informacijske tehnologije. Po opravljenih obveznostih predmeta so študenti izpolnili še anketo, ki je prikazana na sliki Slika 1, odgovori pa so zajeti v Tabela 1.



Slika 1: Anketni vprašalnik, ki so ga izpolnjevali študenti po opravljenih obveznostih predmetov

Sedem vprašanj je bilo zaprtega tipa (pri štirih so bili možni odgovori samo DA in NE, pri treh pa še možnost VSEENO MI JE), devet pa odprtega, ki so predstavljena na sliki Slika 2.

Zakaj se ti (ne) zdi oddajanje nalog s sprotnim obveščanjem smiselno? (npr. "ni čakanja na odgovor asistenta/profesorja", "naloge so prelahke in vedno naredim pravo rešitev v prvo")
Zakaj (ne) bi rad/a videl/a podrobnosti o napakah pri izvajanju sodnika? (npr. "compile error bi mi povedal kje je napaka")
Zakaj se ti (ne) zdi pomembno izvedeti, če je tvoja rešitev zelo ali sploh ni podobna že oddani rešitvi? (npr. "nalogo bi rad/a rešil/a na edinstven način", "važno, da je rešitev pravilna, drugo me ne zanima")
Zakaj (ne) bi reševal/a naloge tudi v kakem drugem programskem jeziku? (npr. "v drugem jeziku znam narediti boljšo rešitev", "drugih jezikov ne poznam")
Katere druge jezike bi rad/a imel/a na voljo?
Zakaj ti tak način oddaje nalog (ne) omogoča dovolj svobode pri reševanju?
Zakaj se (ne) vključuješ v podobna tekmovanja ali prosto reševanje takih problemov?
Od tridesetih (30) vpisanih študentov vas je le sedemnajst (17) oddalo naloge preko spletnega vmesnika v Moodle-u (med temi tudi nekaj plagiatov). Zakaj je bila po tvojem mnenju udeležba tako slaba?

Slika 2: Vprašanja odprtega tipa

4.3 Analiza prostih odgovorov študentov

Večina študentov se strinja, da je tako oddajanje zelo uporabno in smiselno. Pravijo, da je oddajanje domačih nalog s sprotnim obveščanjem o pravilnosti hiter način odpravljanja morebitnih napak, saj ni potrebe po stiku s profesorjem ali asistentom (osebni, elektronska pošta, forum). K temu dodajajo, da bi sporočila prevajalnikov in izvajalnega okolja zelo pripomogla k odpravljanju napak v izdelanih programih. Glede unikatnosti rešitev so razcepljeni na študente, ki bi ta podatek radi izvedeli zaradi same unikatnosti in bi temu primerno poskusili poiskati drugo rešitev, iz splošnega zanimanja in tiste, ki jih ta podatek ne zanima. Ugotovili smo tudi, da del študentov obvlada nekaj drugih programskih jezikov (ML, C++, C# in Python), v katerih bi hitreje znali izdelati boljše, enostavnejše rešitve. Tisti, ki drugih jezikov ne poznajo podpirajo razširitev ponudbe. Velika večina študentov pa se kljub omejitvi programskega jezika, časovni omejitvi in preverjanju plagiatov strinja, da tak način oddajanja nalog omogoča dovolj svobode pri reševanju. Izpostavili bi zanimiv komentar iz manjšine: "Menim, da če moraš kot rešitev naloge dobit točno določeno stvar, potem je vseeno če to rešitev preverja asistent/profesor ali sodnik. Pri reševanju imaš vso svobodo, ki si jo želiš, končni rezultat pa je tako ali tako vnaprej določen". Tovrstna tekmovanja niso najbolj obiskana. Poglavitni razlog je v časovnem primanjkljaju zaradi studijskih obveznosti,

sekundarni razlog pa je v njihovem nepoznavanju. Vsi študenti, za katere se oddajanje nalog ni izšlo zvračajo krivdo na težavnost nalog. Pravijo, da jih zahtevnost nalog primora v "prepisovanje" ali oddajanje napačno rešenih nalog.

	DA				NE				VSEENO			
	P1	P1	P2	PSA	P1	P1	P2	PSA	P1	P1	P2	PSA
Se ti zdi oddajanje nalog s sprotnim obveščanjem smiselno?	11	9	12	12	3	5	3	2	4	4	2	3
Ali bi rad/a videl/a podrobnosti o napakah pri izvajanju sodnika (npr. kaj javi ob Compile error, zakaj Runtime error)?	11	12	11	13	3	3	2	2	4	3	4	2
Ali se ti zdi pomembno izvedeti, če je tvoja rešitev zelo ali sploh ni podobna že oddani rešitvi?	10	9	9	11	5	7	6	3	3	2	2	3
Pri reševanju problemov obstaja več rešitev. Če bi dobil/a podatek o tem, da tvoja rešitev je sicer pravilna ampak ni optimalna, bi jo poskusil/a izboljšati?	4	5	4	3	14	13	13	14				
Sodnik za sedaj sprejema rešitve izdelane le v Javi. Ali bi reševal/a tudi v kakem drugem programskem jeziku (npr. C/ C++, Python, Pascal, Lisp, ML)?	13	15	14	14	5	3	3	3				
Ti tak način oddaje omogoča dovolj svobode pri reševanju?	18	18	17	16	0	0	0	1				
Ali se vključuješ v podobna tekmovanja ali prosto reševanje takih problemov (npr. UVa, SPOJ, UPM)?	4	4	4	5	14	14	13	12				

Tabela 1: Kumulativa odgovorov po predmetih

4.4 Primernost sistema za poučevanje

Sistem za samodejno preverjanje domačih nalog pri poučevanju programiranja se je izkazal kot uporaben. Asistentom oziroma profesorjem omogoča poenostavitev obsežnega dela pregledovanja nalog. Pokazale pa so se tudi določene omejitve, saj lastnosti nalog omejujejo snovalce. Ta problem se še posebej izkaže pri začetnih nalogah, nalogah, ki zahtevajo vnaprej določen način programiranja ter pri nalogah uporabniškega vmesnika. *Začetne naloge* naj bi bile dovolj enostavne in hitro rešljive. Pri tako kratkih izdelkih se poveča verjetnost napačnega označevanja del kot plagiatov, saj je nabor pravilnih rešitev omejen. *Naloge z vnaprej določenim načinom programiranja* zahtevajo izdelavo programov z vnaprej določenim naborom tehnologij oziroma algoritmov. Način preverjanja nalog na osnovi vhoda ter izhoda ne preverja vsebine programov ter tako ne omogoča preverjanja takšnega tipa nalog. Takšna funkcionalnost ostaja na seznamu dodatkov sistema. *Naloge uporabniškega vmesnika* zahtevajo izdelavo programov, ki s svojo funkcionalnostjo ne omogočajo prireditve nalog, kjer bi z izhodnim izpisom opisali pravilnost rešitve. Sistem smo uporabili kot dodatek h “klasičnim” tehnikam preverjanja znanja, h klasičnemu načinu oddaje domačih nalog. V naboru štirih projektnih nalog smo izvedli dve nalogi s pomočjo opisanega sistema, dve nalogi pa je asistent preverjal brez uporabe sistema. Vsak študent je še dodatno izkazal avtorstvo ter osvojeno znanje s kratkim pogovorom o nalogi z asistentom.

5 Diskusija ter nadaljnje delo

Prva izvedba domačih nalog prek opisanega sistema je imela še nekaj zapletov, ki smo jih dokaj hitro odpravili. Pri naslednjih izvedbah se je sistem izkazal za zanesljivega ter enostavnega za uporabo. Izvajalcu predmeta je pri organizaciji takšnega načina oddaje domačih nalog prihranjeno kar nekaj truda, ki ga lahko vloži v dodatne načine preverjanja znanja. Izvajalec mora še vedno preveriti ali je nalogo opravil študent, saj je nadzor nad študenti doma nemogoč. Vendar je takšno preverjanje dokaj preprosto in hitro. Pri predavanju opisanih predmetov na UP FAMNIT bo sistem redno uporabljan kot dodatno orodje in ne bo v popolnosti zamenjal načina preverjanja znanja.

Moodle se je že letos izkazal na tekmovanju za zelo primernega. Za tekmovanje v v naslednjem letu ga bomo nadgradili z modulom za samodejno preverjanje pravilnosti programskih rešitev.

Celoten sistem moramo obravnavati tudi širše. Zaradi svoje tehnike izvedbe ga je mogoče uporabiti tudi v vseživljenjskem izobraževanju, kjer je možnost izvedbe preverjanja znanja (*assesment*) na daljavo še posebej pomembna. Ne nazadnje, sistem je uporaben tudi za potrebe samega izobraževalnega sistema v srednjih in osnovnih šolah ter še posebej kot integralni del mentorskega dela in poučevanja učiteljev (*teacher in-service training*; Istenič et al. 2005, Istenič et al. 2007).

Opombe

1. Leta 1989 je bilo izvedeno tudi prvo mednarodno odprto prvenstvo v znanju računalništva Slovenije, na katerem so sodelovale srednješolci iz petih držav. Tekmovanje je bilo izvedeno v Novi gorici in v organizaciji slovenske Tekmovalne komisije, ki je že pred tem izvajala slovenska prvenstva.

Literatura

Aiken, A. et al. 2006: *MOSS: A system for detecting software plagiarism website*, http://theory.stanford.edu/~aiken/moss/, accessed 25.4.2009.

Booth, S. 2001: *Learning to program as entering the datalogical culture: A phenomenographic exploration*. Proceedings of the 9th EARLI conference.

Bowyer, K. W. and Lawrence O. Hall. 2000: *Experience using »MOSS« to detect cheating on programming assignments*. Department of Computer Science and Engineering, University of South Florida.

Connolly, J. 2008: *ClassEasle: Web 2.0 meets e-learning*. Bachelor of science thesis. Department of Computing Sciences, Villanova University.

Cormack, G. and Graeme Kemkes and Ian Munro and Troy Vasiga. 2006: *Structure, scoring and purpose of computing competition*. University of Waterloo.

Eldering, J. and Thijs Kinkhorst and Peter van de Werken. 2004: *DOMjudge – Programming contest jury system website*, http://domjudge.sourceforge.net/, accessed 25.4.2009.

Garro, A. 2007: *Epaile: Automated grading for computer programming assignments website*, http://epaile.starky.es/, accessed 25.4.2009.

Istenič Starčič, A. et al. 2007: *The Development of the Collaborative Model of ICT Learning System for Lifelong Learning*. WSEAS trans. commun., April 2007, letn. 6, št. 4: 622-627.

Istenič Starčič, A. and Andrej Brodnik. 2005: *Usposabljanje učiteljev za uporabo informacijsko-komunikacijske tehnologije*. Ann, Ser. hist. sociol., 2005, let. 15, št. 1: 163-168.

Kaasboll, J. 1999: *Exploring didactic models for programming*. Tapir: 195-203.

Munda, J. and Mitja Lenič. 2005: *Zaznavanje plagiatov v programskem inženirstvu*. ERK 2005, Book B: 356-59.

Munda, J. and Boris Cigale and Smiljan Šinjur and Mitja Lenič. 2007: *Vpliv avtomatskega preverjanja nalog na izvajanje pedagoškega procesa*. ERK 2007, Book B: 291-94.

Pedroni, M. 2003: *Teaching introductory programming with the inverted curriculum approach*. Bachelor of Science thesis. Department of Computer Science, ETH Zurich.

Tomazin, M. and Andrej Brodnik. 2007: *Učni cilji pouka računalništva v osnovni šoli - slovenski in ACM K12 kurikulum*. Organizacija (Kranj), nov.-dec. 2007, letn. 40, št. 6: A173-A178.