SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1541

IZVEDBA PROGRAMA ZA AUTOMATSKO GENERIRANJE OBRAZACA TESTOVA S PONUĐENIM ODGOVORIMA

Damir Bučar

Zagreb, srpanj 2010.

Zahvaljujem svojoj obitelji na bezuvjetnoj podršci, a posebice mentoru doc.dr.sc. Siniši Šegviću, koji je svojim savjetima i smjernicama pridonio ostvarenju ovog završnog rada.

SADRŽAJ

1 Uvod

Ispiti s ponuđenim odgovorima (eng. *Multiple Choice Exams – MCE's*) često se koriste za vrednovanje znanja studenata. U odnosu na klasična, opisna pitanja, ispravljanje ovakvih provjera znanja višestruko je jednostavnije. Glavni razlog tome je mogućnost objektivnog, konzistentnog i brzog ocjenjivanja [1]. Dodatna prednost je što pri pripremi i ispravljanju takvih vrsta ispita postoji mogućnost korištenja računala [2]. Jedan od sustava razvijenih u tu svrhu je i *Enthusiast* [3].

Javlja se potreba za ostvarenjem novog samostalnog programskog sustava za automatsko generiranje (personaliziranih) obrazaca s ponuđenim odgovorima. Cilj je iskoristiti računalo kao asistenta pri izradi provjera znanja te tako olakšati rad nastavnom osoblju. Pri izradi tog novog sustava, nazvanog *PEGaz*, korištene su tehnologije:

- Programski jezik *Python*
- Sustav za pripremu dokumenata *LaTeX*

U nastavku, nakon opisa programskog sustava *Enthusiast* i kratkog pregleda korištenih tehnologija, slijedi pregled zahtjeva sustava, opis programske izvedbe te prikaz dobivenih rezultata.

2 *Enthusiast*

Enthusiast, razvijen na Fakultetu elektrotehnike i računarstva u Zagrebu, fleksibilan je alat za automatsko generiranje ispita s višestrukim izborom ponuđenih odgovora. U primjeni je od akademske godine 2007/2008., a dosad se koristio na brojnim kolegijima (*Umjetna inteligencija, Interaktivna računalna grafika, Skriptni jezici, Oblikovni obrasci u programiranju, Arhitektura računala 1 i 2*). Koristi se za generiranje ispitnih obrazaca, ubrzanje administracije te u svrhu objektivnog i brzog ocjenjivanja.

Enthusiast se služi bazom podataka i specifikacijom zadanom od strane korisnika kako bi nasumično generirao obrasce spremne za strojno ocjenjivanje. Bazu podataka čini skup tekstualnih datoteka koje sadrže pitanja s ponuđenim odgovorima. Može biti proširena metapodatkovnim oznakama (koje mogu biti hijerarhijski organizirane), kako bi se još uže specificirao odabir zadataka. Varijabilnost između generiranih obrazaca također je moguće podesiti – korištenjem međusobno isključivih varijanti pitanja, kao i redundantnog broja točnih i netočnih odgovora.

Nedostatci *Enthusiast*-a:

- Broj generiranih zadataka i odgovora te naslov zadaće ne mogu se zadati u naredbenom retku (*hardkodirani* su u *LaTeX* predlošku)
- Gornja činjenica povlači nemogućnost generiranja ispisa svih zadataka baze i kontrolu ispravnosti zadataka
- Dio logike delegiran je *LaTeX*-u, kojeg je teže održavati nego *Python* kôd
- Rezultati svakog novog procesiranja (generirana datoteka s točnim odgovorima) gaze rezultate prethodnog
- Na izlazu nije moguće dobiti izravno *.pdf*

Slični razvijeni alati su: *ExamGen, Question Mark, Hot Potatoes, Test Pilot* [3].

3 Korišteni alati

Pri izradi sustava *PEGaz* korišteni su alati *Python* i *LaTeX*, ukratko opisani u nastavku.

3.1 *Python*

Viši programski jezik opće namjene, čiji autor je Guido van Rossum, razvijen je 1990. godine. Sadrži široku i sveobuhvatnu standardnu biblioteku te podržava nekoliko programskih paradigmi – objektno-orijentiranu, imperativnu i funkcijsku. Karakterizira ga potpuna dinamičnost i strogo tipiziranje tipova podataka te automatsko upravljanje memorijom, što ga čini sličnim jezicima *Ruby*, *Perl* i *Scheme*. Kao graničnike blokova kôda koristi uvlačenje linija, što je jedna od bitnih razlika u odnosu na druge popularne programske jezike. Pri dizajnu jezika naglasak je dân čitljivosti kôda, a njegova fleksibilnost, snaga i čista sintaksa čine ga u današnje vrijeme sve popularnijim programskim jezikom [4].

3.2 *LaTeX*

LaTeX je jezik za uređivanje teksta i pripremu dokumenata temeljen na programu *TeX*. Sustav je razvio Leslie Lamport 1980-ih godina, kao skup *TeX* makro naredbi. Koristi se u akademskoj zajednici (matematičari, znanstvenici, inženjeri), kao i u izdavačkim tvrtkama, radi izuzetne kvalitete krajnjeg produkta i širokog spektra mogućnosti oblikovanja sadržaja dokumenata [5][6]. U odnosu na ostale alate za uređivanje teksta (primjerice *MS Word*), *LaTeX* pruža mogućnost automatskog generiranja formulara, a formatiranje teksta je eksplicitno definirano naredbama.

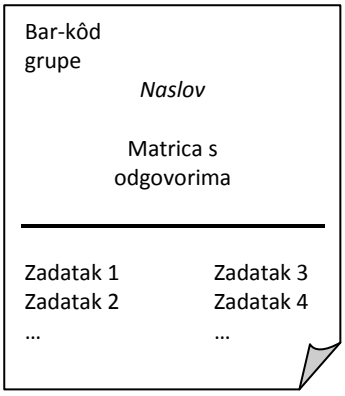
4 Zahtjevi

U sklopu ovog poglavlja opisani su željeni tipovi generiranih dokumenata, način formatiranja tekstualnih datoteka sa zadacima koje čine bazu podataka te podržani načini formatiranja teksta samih zadataka.

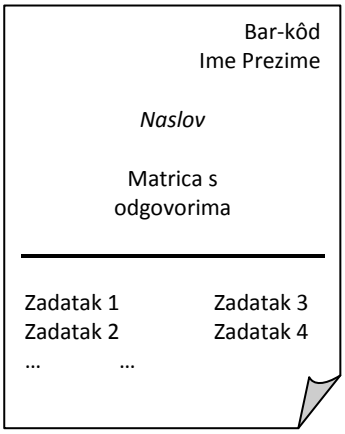
4.1 Slog testova

Kao konačan proizvod sustava *PEGaz*, moguće su tri varijante generiranih dokumenata (obrazaca):

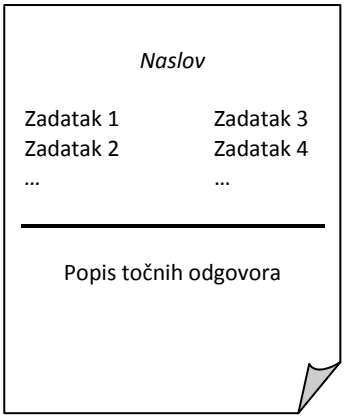
- TIP A - Standardni test
  - Sadrži:
    - kvadratiće za poravnanje (potrebni pri skeniranju obrazaca)
    - bar-kôd grupe
    - naslov
    - matricu s odgovorima
    - dvostupčani ispis zadataka (pitanja s ponuđenim odgovorima)



- TIP B - Personalizirani test
  - Uz elemente standardnog testa, izuzev bar-kôda grupe, sadrži:
    - bar-kôd studenta
    - ime i prezime studenta



- <u>TIP C - Ispis svih zadataka iz zadanog dijela baze podataka</u>
    - Služi kontroli ispravnosti zadataka (pitanja i odgovora) korisnicima sustava - autorima provjerâ znanja
    - Sadrži:
        - naslov (*Ispis zadataka iz direktorija <ime_direktorija>*)
        - dvostupčani ispis zadataka, svaki s oznakom teme i identifikatorom, uz ispis svih ponuđenih odgovora
        - popis točnih odgovora



4.2 Baza podataka

U ovoj verziji podržana je baza podataka (baza zadataka) sačinjena od skupa tekstualnih izvornih datoteka. Format izvornih datoteka i podržani formati teksta samih zadataka opisani su u nastavku.

4.2.1 Formatiranje izvornih datoteka

Podržane su sljedeće leksičke jedinke:

- komentar - *#<komentar>*
- tema zadat(a)ka - *@@<tema>*
- identifikator zadatka (varijanti zadatka) - *@<id_zadatka>*
- tekst zadatka - *<tekst_zadatka>*
    - klasičan tekst, bez prefiksa
    - slijedi nakon identifikatora zadatka, a u slučaju više varijanti istog zadatka nakon praznog retka
    - može se lomiti kroz više redaka
- odgovor - *+/- <tekst_odgovora>*
    - '+' označava točan, '-' netočan odgovor
    - tekst odgovora može se lomiti kroz više redaka

Omogućeno je definiranje više (međusobno isključivih) varijanti jednog zadatka, kojima tada pripada isti identifikator zadatka. Varijante se odvajaju jednom ili više praznih linija.

Primjer izvorne tekstualne datoteke (isječak):

```
...
# pitanja drugog ciklusa predavanja
@@drugi_ciklus_općenito

@7
Za dinamičko mijenjanje postupaka koji se primjenjuju u aplikaciji,
koristimo:
- tvornica
+ strategija
- promatrač
- dekorator
- jedinstveni objekt

# druga varijanta zadatka / iduće pitanje / iduća tema ...
```

4.2.2 Formatiranje zadataka

Tekst pitanja i odgovora u izvornim datotekama može biti formatiran po želji. Sintaksa formatiranja slična je sintaksi *LaTeX*-a, budući se takva sintaksa pokazala praktičnom. Općenita sintaksa formatiranja jest:

\format{formatirani_tekst}.

Kao graničnici pri formatiranju koriste se uređeni parovi znakova, ovisno o kontekstu. Unutar formatiranog teksta drugi član uređenog para ne smije se pojavljivati. Parovi graničnika su:

({ , }), (! , !), ($, $)

Podržani su sljedeći načini formatiranja teksta. Prikazan je način njihovog definiranja u izvornoj datoteci te podržani uređeni par graničnika (u izvornoj datoteci):

- kurziv
  - *\italic{ nakošeno }*
  - *({ , })*

- podebljano
  - *\bold{ važno }*
  - *({ , })*

- matematička formula
  - u retku s tekstom (eng. *inline*)
    - *\math{ a = b +2 }*
  - kroz čitavi redak
    - *\eq{ a^2 = b^2 + c^2 }*
  - bilo koji par graničnika

- izvorni kod
  - u retku s tekstom (eng. *inline*)
    - *code{ a = 2; }*
  - programski isječak kroz više redaka
    - *codeblock{*

      if (a > 2):

      a = a – 1

      print a

      }
  - bilo koji par graničnika, osim u slučaju kad se u sklopu formatiranog teksta želi koristiti vitičaste zagrade – tad isključivo (*!* , *!*) ili (*$* , *$*)

- latex
  - za formate koji nisu podržani, a postoji potreba za korištenjem postojećih u sustavu *LaTeX*, omogućeno je formatiranje sljedećeg oblika:
    - *latex!<izravan_zapis_u_latexu>!*
  - (*!* , *!*) ili (*$* , *$*)

4.3 Izvedba formatiranja u *LaTeX*-u

Kako bi se razlikovale od običnog teksta, naredbe *LaTeX*-a predznačuju se silaznom kosom crtom (eng. *backslash*, "\"). Kod naredbi koje primaju argumente, argumenti se smještaju unutar graničnika (npr. vitičaste zagrade) [7].

4.3.1 Specijalni simboli LaTeX-a

Obzirom da "\" za *LaTeX* ima posebno značenje, u konačnom dokumentu ne može se dobiti izravnim unosom. Uz obrnutu kosu crtu, postoji još 9 specijalnih simbola *LaTeX*-a [8]. Stoga im treba posvetiti posebnu pozornost pri obradi izvornih datoteka i generiranju konačnih dokumenata. Iz naše izvorne datoteke u kojoj su zapisani, prilikom prevođenja zapisa u *.tex* datoteku, treba ih nekako predznačiti, tj. prevesti u naredbe koje će *LaTeX* prepoznati i generirati odgovarajuće simbole.

Tablica 1: Specijalni simboli *LaTeX*-a i pripadne naredbe koje ih generiraju

Simbol	Naredba
~	\textasciitilde
#	\#
$	\$
%	\%
^	\textasciicircum
&	\&
_	_
\	\textbackslash
{	\{
}	\}

4.3.2 Preslikavanje PEGaz → LaTeX

U poglavlju *4.2.2 Formatiranje zadataka* navedeni su podržani formati teksta u sklopu izvornih datoteka sustava *PEGaz*. U *Tablici 2* prikazano je kako se ti formati preslikavaju u pripadne formate sustava *LaTeX*.

Tablica 2: Preslikavanje formata *PEGaz - LaTeX*

PEGaz	***LaTeX***
\italic{ *<formatirani_tekst>* }	\emph{ *<formatirani_tekst>* }
\bold{ *<formatirani_tekst>* }	\textbf{ *<formatirani_tekst>* }
\math{ *<matematička_formula>* }	$ *<matematička_formula>* $
\eq{ *<matematička_formula>* }	$$ *<matematička_formula>* $$
\code{ *<isječak_kôda>* }	\lstinline{ *<isječak_kôda>* }
\codeblock{ *<isječak_kôda>* }	\begin{lstlisting} *<isječak_kôda>* \end{lstlisting}
\latex{ *<nativni_latex>* }	*<nativni_latex>*

5 Programska izvedba

Veći dio programske logike sustava *PEGaz* prepušten je modulima *Python*-a. U sklopu ovog poglavlja opisana je programska izvedba sustava. Dân je pregled strukture i bitnih funkcija paketa i pojedinih modula *Python*-a te predložaka *LaTeX*-a korištenih pri generiranju dokumenata.

5.1 Paketi i moduli sustava *PEGaz*

Jezgra programskog sustava *PEGaz* sačinjena je od paketa i pripadnih modula opisanih u nastavku.

Main.py

- Vršni modul koji kroji tok podataka čitavog sustava, ne pripada niti jednom paketu
- Slijedno učitava module prethodno postavljajući parametre potrebne za njihovo ispravno izvođenje

Paket optargs

OptArgs.py

- Obrađuje argumente komandne linije (korisničku specifikaciju) i pohranjuje ih u prikladne strukture podataka

Paket reader

FileReader.py

- Automat parser - obrađuje izvorne datoteke iz zadanog direktorija
- Učitava zadatke iz datotekâ i pohranjuje ih u memoriju - u objekte klase *Question* sadržane u modulu *Question.py*
- Koristi modul *States.py*

States.py

- Imitacija stanja automata modula *FileReader.py*, svojevrsna enumeracija
- Stanja (odgovaraju elementima gramatike): *IDLE, THEME, QID, QTEXT, ANSWER*

Question.py

- Klasa *Question*
    - Članske varijable – *theme, qID, qText, answers, posAnswers, negAnswers*

Paket latexGen

LTXGenerator.py

- Generator *.tex* datoteka, jedan od ključnih modula sustava
- Koristi module *Selector.py*, *Tokenizer.py* te *LTXTypeSetter.py*
- Bitne metode:
    - *generateQuestions ()*
        - iz odabranih varijanti zadataka generira njihove zapise u *LaTeX*-u
        - generiranje redoslijeda odgovora odvija se nasumično – pokušava se još više razlikovati obrasce testova
    - *generatePositiveAnswers ()*
        - kreira popis točnih odgovora na kraju dokumenta (ukoliko je odabran oblik dokumenta tipa C)
    - *generateAnswersMatrix ()*
        - generira zapis matrice odgovora (kod dokumenata tipa A i B)
    - *printToFile(i)*
        - ovisno o identifikatoru *i*, vrši konačni zapis u ciljnu *.tex* datoteku

LTXTypeSetter.py

- Sadrži rječnik podržanih formata sustava *PEGaz* i ekvivalenata u sustavu *LaTeX*, na osnovu kojeg obavlja pretvorbu formata *PEGaz* →*LaTeX*
- Koristi modul *LTXEscaper.py*, točnije njegovu metodu *doEscape*, a poziva je samo nad neformatiranim (*plain*) dijelovima teksta
- Bitne metode:
    - *doReformat (txtObjList)*
        - iz liste objekata tipa *TextObject* kreira niz znakova (*string*) spreman za pohranu u *.tex* datoteku i vraća ga kao povratnu vrijednost

LTXEscaper.py

- Sadrži rječnik specijalnih simbola *LaTeX*-a na osnovu kojeg dotične simbole prevodi u naredbe sustava *LaTeX*
- Bitne metode:
    - *doEscape (string)* – nad primljenim nizom znakova vrši zamjenu specijalnih simbola *LaTeX*-a ekvivalentnim generirajućim naredbama

Paket selector

Selector.py

- Vrši odabir zadataka prema zadanim parametrima
- Bitne metode:
    - *doSelection()*
        - kreira i vraća listu odabranih varijanti zadataka
    - *selectVariants (num, variants)*
        - nasumično odabire *num* varijanti zadataka iz liste *variants*
        - ne odabire više od jednog zadatka s istim identifikatorom (na istom obrascu ne mogu se naći dvije varijante istog zadatka) – varijante su međusobno isključive
    - *selectAnswers (num, ansList)*
        - nasumično odabire *num* odgovora iz liste *ansList*

Paket tokenizer

Tokenizer.py

- Svojevrsni leksički analizator – odsječcima teksta pridjeljuje oznake formata
- Isječci se pohranjuju u objekte klase *TextObject* sadržane u modulu *TextObject.py*
- Bitne metode:
    - *doTokenize (objList, string)*
        - zadani niz znakova (*string*) tokenizira, a objekte pohranjuje u listu *objList*

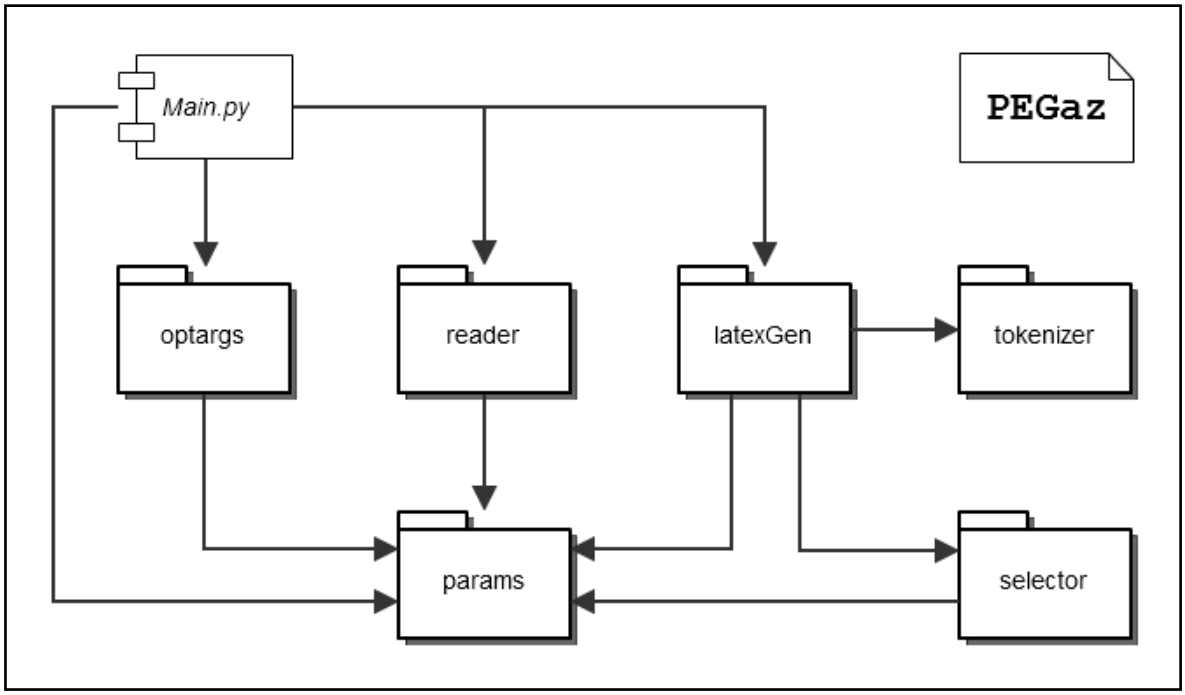
TextObject.py

- Klasa *TextObject*
    - Članske varijable – *text, textformat, startDelimiter, endDelimiter*

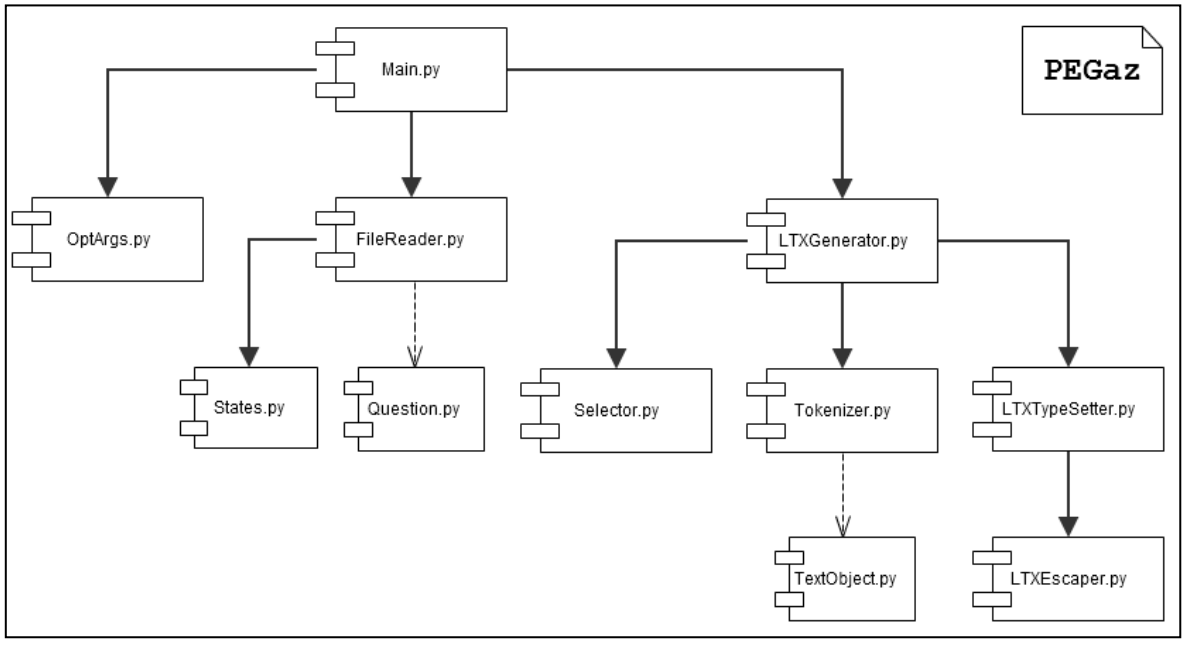
Paket params

OptArgs_params.py, FileReader_params.py, LTXGenerator_params.py, Selector_params.py

Moduli ovog paketa sadrže podatke koji predstavljaju parametre preostalih modula.



Slika 1: Dijagram ovisnosti paketa sustava *PEGaz*



Slika 2: Dijagram ovisnosti modula (radi preglednosti nisu prikazani moduli paketa params)

5.2 *LaTeX* predlošci

Kreirana su 3 predloška u *.tex* formatu. Korisnik opcijama specificira izbor predloška koji definira konačnu prezentaciju, tj. izgled generiranog dokumenta.

Od važnijih *LaTeX* paketa, svi predlošci koriste:

- `multicols` – ispis zadataka u dvostupčanom formatu
- `enumerate` – enumeracija odgovora (pitanja se numeriraju „ručno“ u sklopu *Python* skripte)
- `listings` – za eventualnu upotrebu formata programskog kôda

Također, za podršku hrvatskih dijakritičkih znakova, zadaju se naredbe:

```
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

Svaki od predložaka odgovara jednom od tipova obrazaca definiranih u zahtjevima sustava:

- Za TIP A – predložak *standardTest.tex*
  - definirana naredba `\answersMatrix` u sklopu koje se nalazi oznaka *&* - slijedi generiranje matrice odgovora
  - za potrebe generiranja bar-kôda grupe koristi pakete `pstricks` te `pst-barcode`
  - koriste se i paket `xy` te naredba `\xymatrix` za potrebe generiranja matrice
- Za TIP B – predložak *personalizedTest.tex*
  - uz karakteristike obrasca tipa A, koristi pakete `pstricks` te `pst-barcode` za potrebe generiranja bar-kôda studenta
- Za TIP C – predložak *printAll.tex*
  - najjednostavniji od svih predložaka
  - u sklopu naredbe `\begin{multicols}{n}...\end{multicols}` nalaze se oznake *#* i *@* za generiranje zadataka (*n=2*) i generiranje popisa točnih odgovora (*n=4*)

Obradu predloška u svrhu generiranja konačnog dokumenta obavlja modul *LTXGenerator.py*, opisan u prethodnom potpoglavlju. Nailaskom na liniju s jednim od definiranih uzoraka započinje generiranje dijelova konačnog dokumenta:

- *#* : generiraj **zadatke** (odabrana pitanja i pripadne ponuđene odgovore)
- *&* : generiraj **matricu s odgovorima** – dimenzije određuje broj odabranih zadataka i ponuđenih točnih i netočnih odgovora (ne koristi se kod predloška *printAll.tex*)
- *@* : generiraj **popis točnih odgovora** za ispisane zadatke (koristi se samo kod predloška *printAll.tex*)

Primjer predloška – *printAll.tex* (isječak):

```
...
\begin{document}

\begin{center}
\textbf{\large \textsf{Ispis zadataka iz direktorija \dirName}}
\end{center}

\vspace{1cm}

\begin{multicols}{2}
#
\end{multicols}

\vspace{1cm}

\rule{\mywidth}{0.5mm}
\begin{center}
\textbf{Popis točnih odgovora}
\end{center}

\begin{multicols}{4}
@
\end{multicols}

\newpage


\end{document}
```

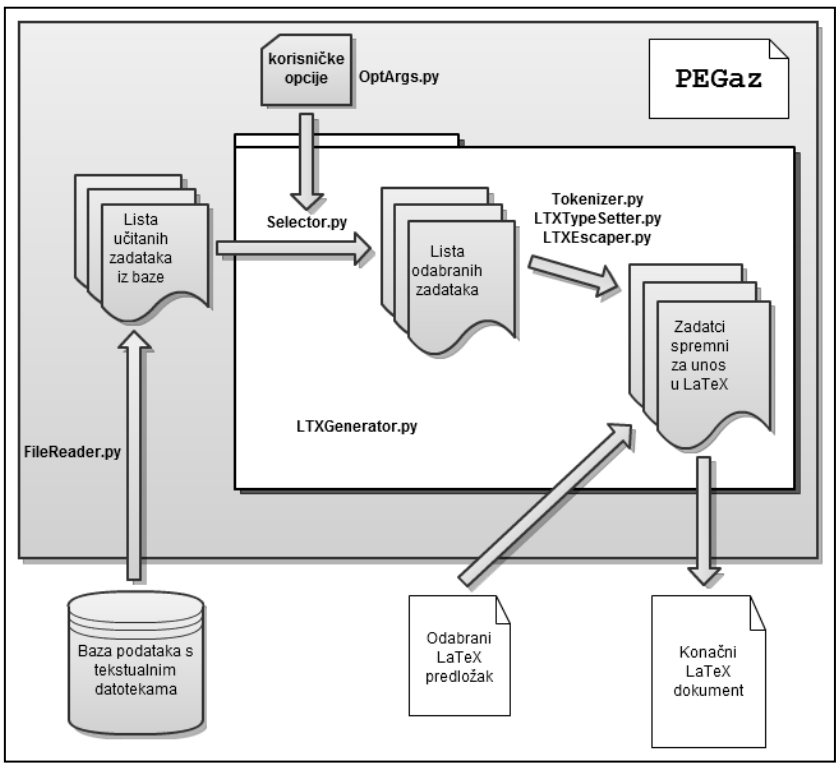
5.3 Tok podataka – pregled interakcija čitavog sustava

Korisnik prilikom pokretanja programa zadaje specifikaciju prema kojoj sustav treba generirati konačan dokument – tip obrasca, naslov, broj pitanja i odgovora itd.

Modul *Main* upravlja čitavim procesom prikazanim na *Slici 3*.

1. Modul *OptArgs* dohvaća korisničke opcije, koje bivaju proslijeđene modulima *Selector* i *LTXGenerator.*
2. Modul *FileReader* potom učitava sve zadatke iz zadanog direktorija (dijela baze podataka).
3. Zatim se pokreće modul *LTXGenerator*, koji sadrži najveći dio funkcionalnosti čitavog sustava. Pri svom radu koristi module:
    a. *Selector* – odabir zadataka
    b. *Tokenizer* – svojevrsna leksička analiza (tokenizacija) teksta pitanja i odgovora
    c. *LTXTypeSetter* i *LTXEscaper* – pretvorba formata *PEGaz* → *LaTeX*

U konačnici, pomoću izabranog predloška, modul generira od korisnika specificiran obrazac (konačnu *.tex* datoteku).



Slika 3: Pregled interakcija sustava *PEGaz*

6 Primjeri upotrebe i rezultati

U sklopu ovog poglavlja opisani su primjeri upotrebe programskog sustava *PEGaz*, odnosno načini pozivanja programa iz komandne linije (ulaz programa) te generirani dokumenti (izlaz programa).

6.1 Primjeri upotrebe

PEGaz se poziva naredbama komandne linije, uz navedene podržane (kratke i duge) opcije:

- *-a* ili *--all*
  - odabir svih zadataka iz zadanog dijela baze podataka (direktorija)
- *-q* ili *--questions_directory*
  - izvor zadataka
- *-s* ili *--selection*
  - odabir tema i broja zadataka iz pojedine teme
- *-p* ili *--positive_answers*
  - broj točnih odgovora
- *-n* ili *--negative_answers*
  - broj netočnih odgovora
- *-t* ili *--title*
  - naslov obrasca
- *-i* ili *--person_info*
  - datoteka s osobnim podacima studenata (za obrazac tipa B)
- *-g* ili *--group_id*
  - identifikator (početne) grupe i broj željenih grupa (za obrazac tipa A)

Sve opcije osim prve imaju i pripadne argumente.

Primjer upotrebe dân je za svaki primjer željenog ciljnog dokumenta:

1. **TIP A)** Ispitni obrazac s 2 zadatka iz cjeline *biljke* i 2 zadatka iz cjeline *životinje*. Svako pitanje ima 1 točan i 3 netočna ponuđena odgovora. Identifikator prve grupe je 1001, a broj grupa je 4. Naslov provjere znanja je *"Ulazni blic 2.LV"*, a zadatci se preuzimaju iz direktorija *dir1*.

   $ pegaz –s biljke:2 –s životinje:2 –p 1 –n 3 –g 1001:4 –t "Ulazni blic 2.LV" –q dir1

2. **TIP B)** Skup personaliziranih ispitnih obrazaca s 2 zadatka iz cjeline *biljke* i 2 zadatka iz cjeline *životinje*. Svako pitanje ima 1 točan i 3 netočna odgovora. Naslov provjere je *"Ulazni blic 2.LV"*. Zadatci se preuzimaju iz direktorija *dir1*, a podatci o studentima iz datoteke *info.txt*.

   $ pegaz –s biljke:2 –s životinje:2 –p 1 –n 3 –t "Ulazni blic 2. LV" –q dir1 –i info.txt

3. **TIP C)** Dokument s ispisom svih zadataka iz svih datoteka direktorija *c:/testovi2*. Popis točnih odgovora nalazi se na zadnjoj stranici dokumenta.

   $ pegaz –a –q c:/testovi2

Omogućen je i alternativan poziv programa, gdje su svi argumenti zadani u konfiguracijskoj datoteci, označeni nazivima dugih ili kratkih opcija. Poziv programa u tom slučaju:

$ pegaz config.txt

Konfiguracijska datoteka (*config.txt*) ima sljedeći oblik (poziv jednak onome pod točkom 2.):

```
questions_directory: dir1
selection: biljke:2
selection: životinje:2
positive_answers: 1
negative_answers: 3
title: Ulazni blic 2.LV
person_info: info.txt
```

Po završetku generiranja *.tex* datoteka, kako bi se dobio konačni *PDF* dokument, potrebno je pokrenuti sljedeći niz naredbi:

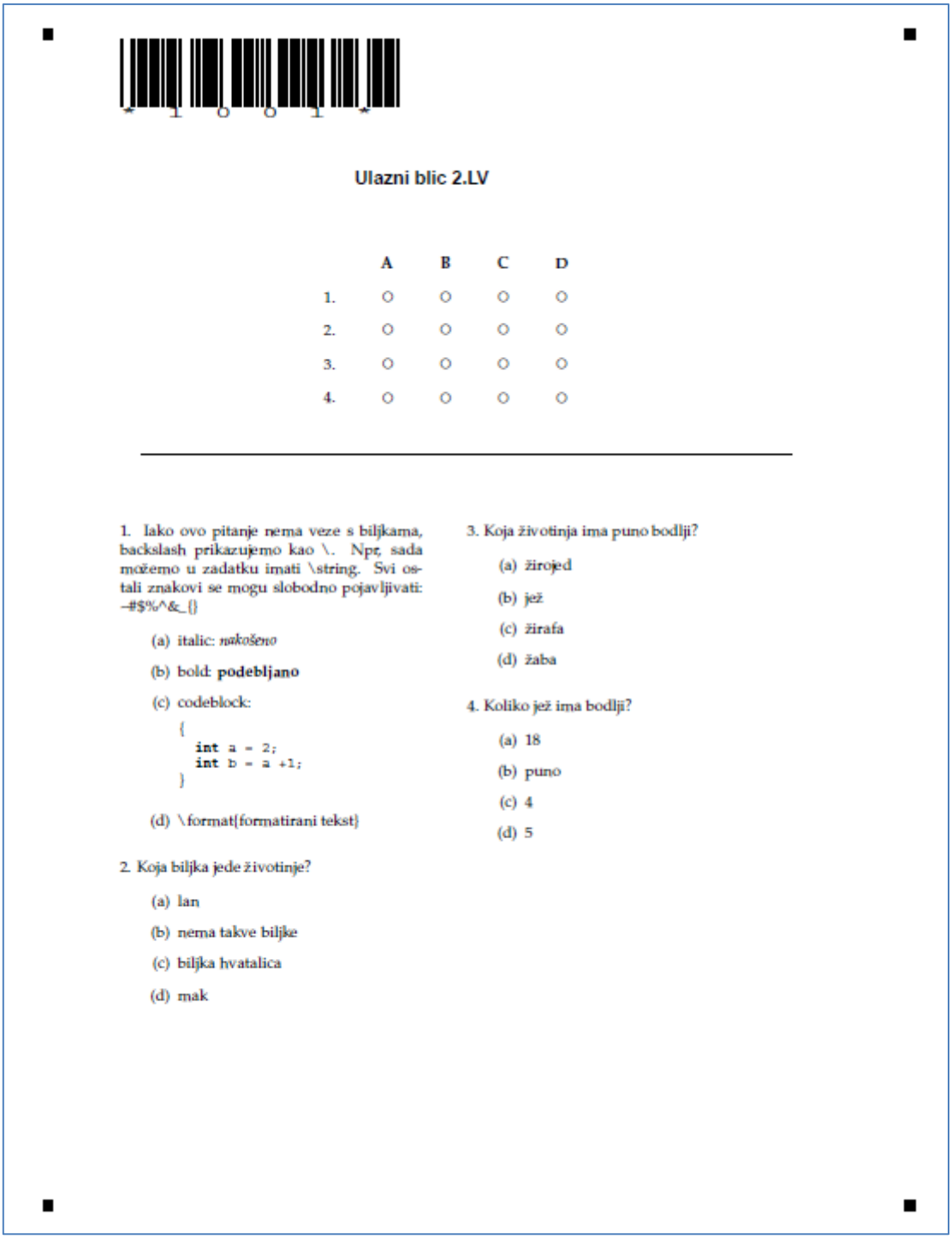
- *latex <naziv_datoteke>.tex*
- *dvips <naziv_datoteke>.dvi*
- *ps2pdf <naziv_datoteke>.ps*

U slučaju da se koristi varijanta obrasca koji ne sadrži bar-kôd (TIP C), moguć je i jednostavniji poziv koji izravno iz *.tex* datoteke generira *PDF* dokument:

- *pdflatex <naziv_datoteke>.tex*

6.2 Rezultati

Dokumenti generirani ranije navedenim primjerima korištenja programa prikazani su u nastavku.



1001

Ulazni blic 2.LV

	A	B	C	D
1.	O	O	O	O
2.	O	O	O	O
3.	O	O	O	O
4.	O	O	O	O

1. Iako ovo pitanje nema veze s biljkama, backslash prikazujemo kao \. Npr, sada možemo u zadatku imati \string. Svi ostali znakovi se mogu slobodno pojavljivati: -#$%^&_{}

   (a) italic: *nakošeno*

   (b) bold: **podebljano**

   (c) codeblock:

```
{
  int a = 2;
  int b = a +1;
}
```

   (d) \format{formatirani tekst}

2. Koja biljka jede životinje?

   (a) lan

   (b) nema takve biljke

   (c) biljka hvatalica

   (d) mak

3. Koja životinja ima puno bodlji?

   (a) žirojed

   (b) jež

   (c) žirafa

   (d) žaba

4. Koliko jež ima bodlji?

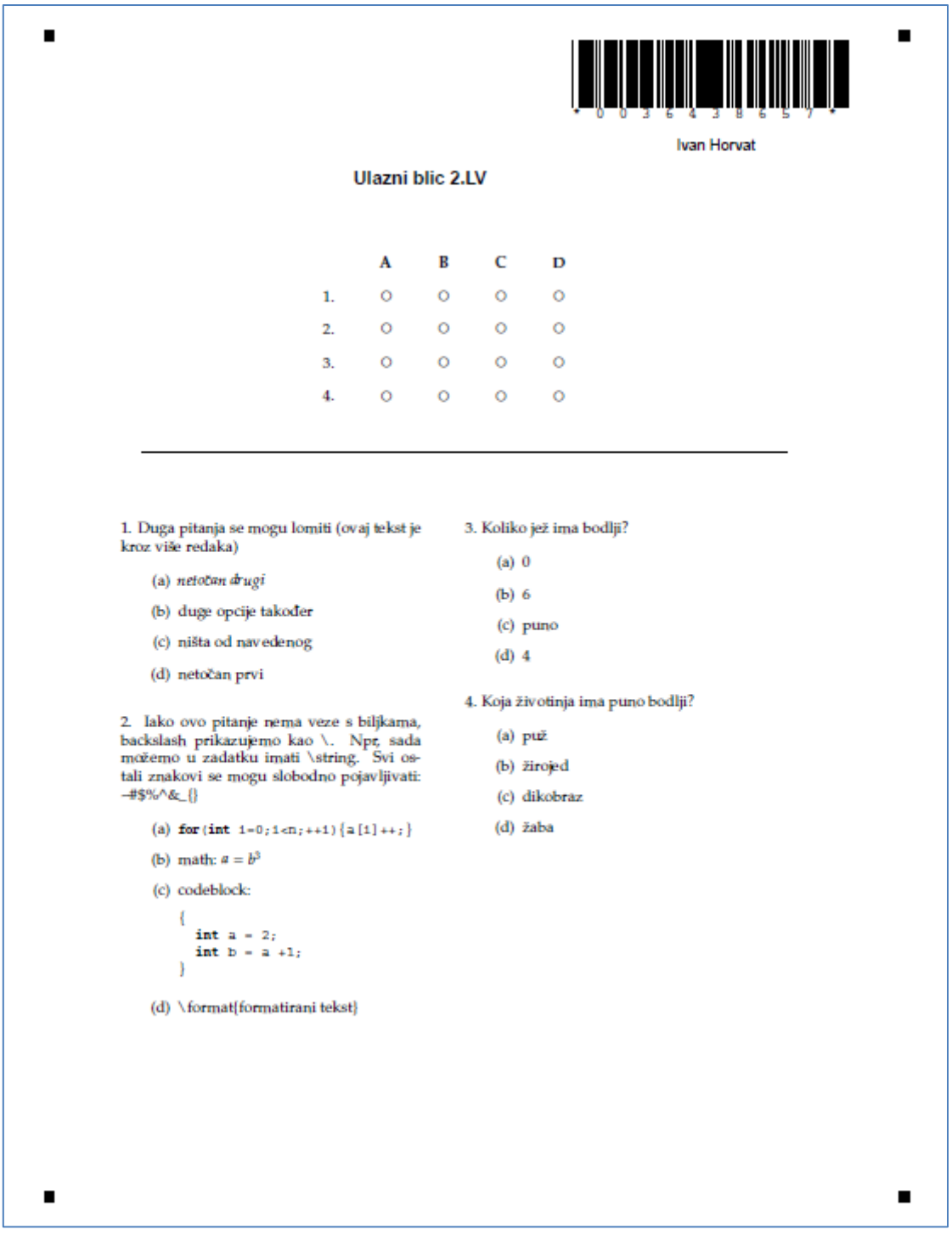
   (a) 18

   (b) puno

   (c) 4

   (d) 5

Slika 4: TIP A – standardni test



Ivan Horvat

Ulazni blic 2.LV

	A	B	C	D
1.	O	O	O	O
2.	O	O	O	O
3.	O	O	O	O
4.	O	O	O	O

1. Duga pitanja se mogu lomiti (ovaj tekst je kroz više redaka)

   (a) *netočan drugi*

   (b) duge opcije također

   (c) ništa od navedenog

   (d) netočan prvi

2. Iako ovo pitanje nema veze s biljkama, backslash prikazujemo kao \. Npr, sada možemo u zadatku imati \string. Svi ostali znakovi se mogu slobodno pojavljivati: –#$%^&_{}

   (a) `for(int i=0;i<n;++i){a[i]++;}`

   (b) math: $a = b^3$

   (c) codeblock:

```
{
  int a = 2;
  int b = a +1;
}
```

   (d) \format{formatirani tekst}

3. Koliko jež ima bodlji?

   (a) 0

   (b) 6

   (c) puno

   (d) 4

4. Koja životinja ima puno bodlji?

   (a) puž

   (b) žirojed

   (c) dikobraz

   (d) žaba

Slika 5: TIP B – personalizirani test

Ispis zadataka iz direktorija c:/testovi2

1. životinje - 1000 - Koliko jež ima bodlji?

   (a) 0

   (b) 5

   (c) 18

   (d) 6

   (e) puno

   (f) 4

2. životinje - 1001 - Koja životinja ima puno bodlji?

   (a) žirojed

   (b) žirafa

   (c) dikobraz

   (d) žaba

   (e) jež

   (f) puž

3. biljke - 2000 - Koja biljka jede životinje?

   (a) lan

   (b) mak

   (c) biljka hvatalica

   (d) poriluk

   (e) nema takve biljke

4. biljke - 2000 - Duga pitanja se mogu lomiti (ovaj tekst je kroz više redaka)

   (a) **točan odgovor**

   (b) *netočan drugi*

   (c) netočan prvi

   (d) ništa od navedenog

   (e) duge opcije također

5. biljke - 2010 - Iako ovo pitanje nema veze s biljkama, backslash prikazujemo kao \. Npr, sada možemo u zadatku imati \string. Svi ostali znakovi se mogu slobodno pojavljivati: ~#$%^&_{}

   (a) italic: *nakošeno*

   (b) bold: **podebljano**

   (c) codeblock:

```
{
  int a = 2;
  int b = a +1;
}
```

   (d) **for** `(int i=0; i<n; ++i) {a[i]++;}`

   (e) math: $a = b^3$

   (f) eq:

$$c^2 = a^2 + b^2$$

   (g) \format{formatirani tekst}

Popis točnih odgovora

1. - E
2. - C, E
3. - C
4. - A, D
5. - G

1

Slika 6: TIP C – ispis svih zadataka direktorija

7 Zaključak

Pri oblikovanju i razvoju sustava *PEGaz* osnovni cilj bio je stvoriti kompaktan, za održavanje i korištenje jednostavan proizvod programske potpore. Kao i kod ostalih sustava za automatsko generiranje ispitnih obrazaca, ciljana korisnička skupina jest prvenstveno nastavno osoblje.

Korisničkom specifikacijom ulaznih parametara definira se konačan dokument – jedan od tri predefinirana tipa obrazaca. Eliminirana je potreba za ručnim odabirom zadataka (pitanja i ponuđenih odgovora) te ručnim formatiranjem teksta. Omogućeno je i automatsko ispravljanje obrazaca, budući su obrasci tipa A i B spremni za skeniranje i strojnu analizu. Posebice koristan je i obrazac tipa C, koji omogućuje pregled i kontrolu ispravnosti zadataka iz baze podataka. Tim činjenicama bitno je olakšan posao nastavnom osoblju pri pripremi, ali i ispravljanju ispita.

Prilikom izvedbe trenutne verzije programa esencijalni dijelovi komponenata *Python*-a napisani su kao klasične skripte (koje se učitavanjem izvode slijedno), dok se objektni pristup primijenio tek na pohranu podataka u vlastito definirane razrede. Dio razloga tomu leži i u činjenici da se autor po prvi puta susreo s tehnologijama u kojima je sustav *PEGaz* razvijen – *Python* i *LaTeX*.

U sklopu potencijalnih budućih verzija moguće je prekrajanje programskog kôda i organizacije u vidu poboljšanja i potpune objektne orijentiranosti čitavog sustava. Također, planira se i dodavanje dodatnih funkcionalnosti, kao što su, primjerice:

- mogućnost jednostavnog uključivanja slikâ u sklopu pojedinog zadatka (trenutno moguće korištenjem oznake formata *latex* i naredbom u nativnom *LaTeX*-u)
- pridjeljivanje metapodatkovnih oznaka (eng. *tags*) zadatcima, kako bi se dodatno pospješio odabir i filtriranje zadataka (analogno sustavu *Enthusiast* [3]).

Konačno, sustav je spreman podržati eventualne nove tehnologije za pripremu dokumenata, umjesto trenutno podržanog *LaTeX*-a.

8 Literatura

[1] Čupić, M.; Šnajder, J.; Dalbelo Bašić, B.: *Post-test analysis of automatically generated multiple choice exams: a case study*, Conference ICL2009, Villach, Austria, 2009

[2] Čupić, M.: *A case study: using multiple-choice tests at university level courses – preparation and supporting infrastructure*

[3] Šnajder, J.; Čupić, M.; Dalbelo Bašić, B.; Petrović, S.*: Enthusiast: An authoring tool for automatic generation of paper-and-pencil multiple-choice tests*, Conference ICL2008, Villach, Austria, 2008

[4] http://www.python.org, travanj 2010.

[5] http://www.latex-project.org/, svibanj 2010.

[6] Lamport, L.: *LaTeX: A Document Preparation System*, http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html, lipanj 2010.

[7] *Text Formatting with LATEX, A Tutorial*, Academic and Research Computing, April 2007, http://www.rpi.edu/campus/doc/acs.memos/rpi109.pdf

[8] Krishnan, E.; Krishna, G.S.: *LaTeX Tutorials – A Primer*, Indian TeX Users Group, 2003, http://www.eng.cam.ac.uk/help/tpl/textprocessing/ltxprimer-1.0.pdf, lipanj 2010.

9 Sažetak / Abstract

IZVEDBA PROGRAMA ZA AUTOMATSKO GENERIRANJE OBRAZACA TESTOVA S PONUĐENIM ODGOVORIMA

Ispiti s ponuđenim odgovorima često se koriste za vrednovanje znanja studenata. Zbog nedostataka postojećih rješenja javila se potreba za ostvarenjem novog programskog sustava za automatsko generiranje obrazaca testova s ponuđenim odgovorima. U okviru ovog rada takav sustav je ostvaren pomoću programskog jezika *Python* i sustava za uređivanje dokumenata *LaTeX*. Razvijeni sustav *PEGaz* koristi bazu zadataka sačinjenu od tekstualnih datoteka te korisničku specifikaciju koja definira konačan dokument. Omogućeno je generiranje tri tipa obrazaca – standardni test, personalizirani test (prikladni za strojno ocjenjivanje) te dokument s ispisom svih zadataka iz zadane baze podataka (prikladan za kontrolu ispravnosti zadataka).

Ključne riječi: vrednovanje znanja, testovi s ponuđenim odgovorima, *PEGaz*, *Python*, *LaTeX*

DESIGN AND IMPLEMENTATION OF A PROGRAM FOR GENERATING EXAMS WITH MULTIPLE-CHOICE QUESTIONS

Multiple-choice exams (MCE's) are often used for students' knowledge assessment. Due to the shortcomings of existing software solutions, a need emerged to design and implement a new software for automatic generation of test sheets with multiple-choice questions. This paper describes the *PEGaz* system, implemented using *Python* programming language and *LaTeX* – a document preparation system. The system uses a questions database made of plain-text files and user specification which defines the final document. *PEGaz* generates one of the three types of documents – a standard test, a personalized test (suitable for machine-scoring) and a document which contains all of the questions from the given database (suitable for questions' validation).

Key words: knowledge assessment, multiple-choice exams, *PEGaz*, *Python*, *LaTeX*