Parameterized and automated assessment on an introductory programming course

Francisco de Assis Zampirolli, Paulo Henrique Pisani,
João Marcelo Borovina Josko, Guiou Kobayashi,
Francisco J. Fraga, Denise Hideko Goya, Heitor Rodrigues Savegnago

Universidade Federal do ABC, Brasil

Simpósio Brasileiro de Informática na Educação
Natal, 2020

Contents

Motivation

- How to generate exams for many students?
  - Using a web platform with database of questions
  - Dedicated to Education Systems
- How to minimize fraud?
  - An exam in which the questions are unique to each student.
- How to correct this exam automatically?
  - Using Moodle with VPL plugin.

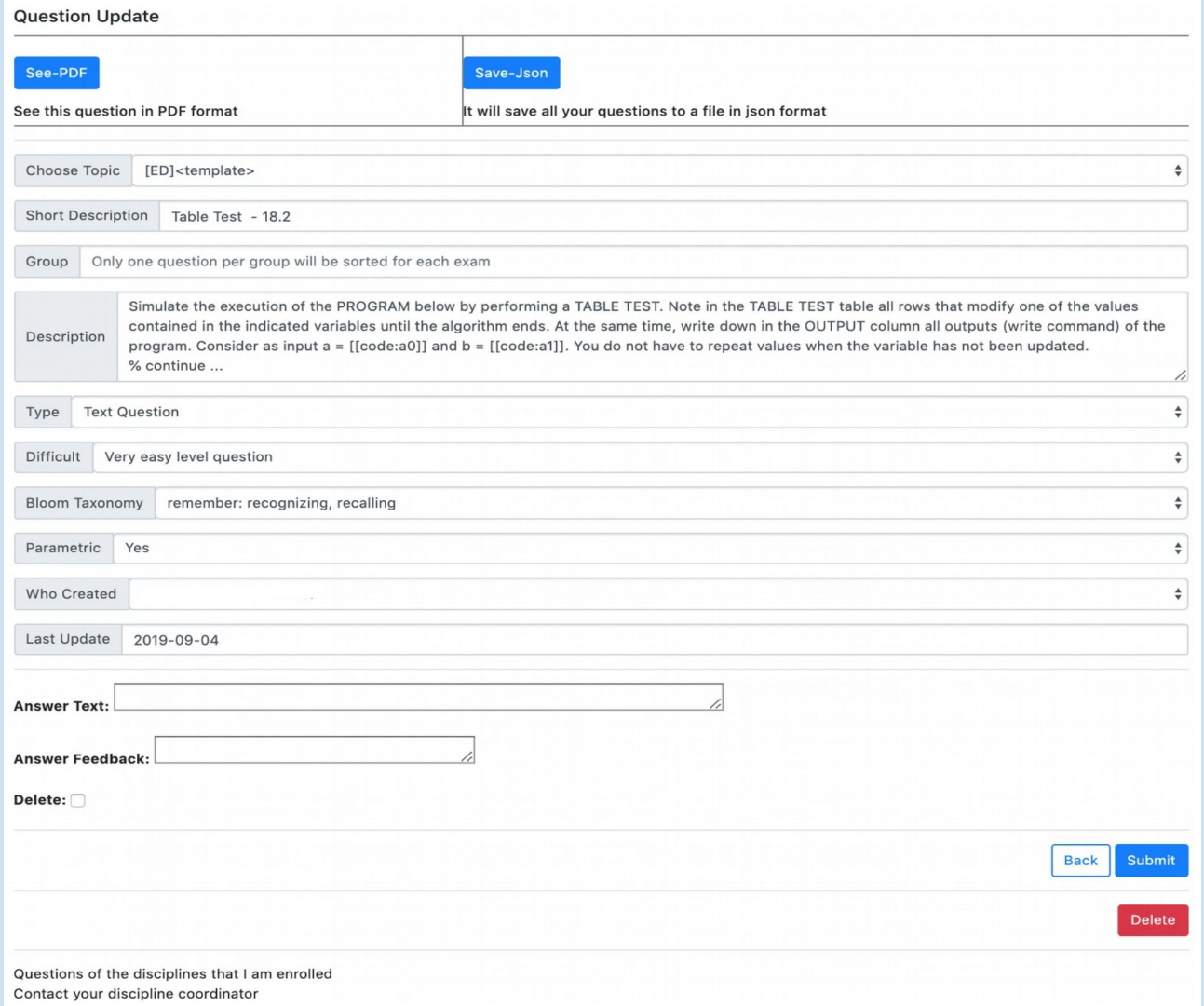
Contents

- Motivation
- Materials and methods
  - Questions to Discussion
    - **Assessment with manual correction**
    - Assessment with code submission on the computer
  - Proposal to integrate MCTest and VPL – applied in 2019
- Preliminary results and discussions
- Conclusion and Future Works

Method:

Create question,

Manual work on Paper



Method: Create question, **Manual work on paper**

#1238 1. Simulate the execution of the PROGRAM below by performing a TABLE TEST. Note in the TABLE TEST table all rows that modify one of the values contained in the indicated variables until the algorithm ends. At the same time, write down in the OUTPUT column all outputs (write command) of the program. Consider as input a = 16 and b = 19. You do not have to repeat values when the variable has not been updated.

```
program { fuction begin() {
1   integer a=-1, b=-2, c=5, d=4
2   read(a)
3   read(b)
4   while (d>0) {
5     d=d-1
6     if (b<a) {
7       a=a-1
8       write("\n111")
9     }
10    if (b>a) {
11      write("\n222")
12    } else {
13      b=b+2
14      write("\n333")
15    }
16 }
}}
```

TABLE TEST					
row	a	b	c	d	OUT.

Method:
Create question,
Manual work on Paper

```
% Part I: description of the question
Simulate the execution of the PROGRAM below by performing a
TABLE TEST. Note in the TABLE TEST table all rows that
modify one of the values contained in the indicated variables until
the algorithm ends. At the same time, write down in the OUTPUT
column all outputs (write command) of the program. Consider as
input a = [[code:a0]] and b = [[code:a1]]. You do not have to repeat
values when the variable has not been updated.

% Part II: table
\newcolumntype{C}{>{\centering\arraybackslash}p{3.1em}}
\begin{multicols}{2}
\begin{lstlisting}

% Part II.a: program to be simulated - Left Column
program { function begin() {
1  integer a=-1, b=-2, c=[[code:a2]], d=[[code:a3]]
2  read(a)
3  read(b)
4  while (d>0) {
5    d=d-1
6    if (b<a) {
7      a=a-[[code:a4]]
8      write("\n111")
9    }
10   if (b>a) {
11     write("\n222")
12   } else {
13     b=b+[[code:a5]]
14     write("\n333")
15   }
16 }
}}
\end{lstlisting}
\columnbreak

% Part II.b: location of Student Response - Right Column
\centering
\begin{tabular}{|l|C|C|C|C|C|c|}
\hline
\multicolumn{6}{|c|}{\textbf{TABLE TEST}} \\ \hline
row   & a   & b   & c   & d  & OUT. \\ \hline
      &     &     &     &    &      \\ \hline
% ...
\end{tabular}
\end{multicols}
\newpage

% Part III: the correct answer on the back of the sheet
{\color{bubbles}
\begin{verbatim}
[[code:mySimulation]]
\end{verbatim}
}

% Part IV: python code between "[[def:" and "]]"
[[def:
import random

# Part IV.a: function to create random variables
def myRandom():
 global a0,a1,a2,a3,a4,a5
 a0=random.randrange(11, 20, 1)
 a1=random.randrange(11, 20, 1)
 a2=random.randrange(5, 7, 1)
 a3=random.randrange(1, 7, 1)
 a4=random.randrange(1, 4, 1)
 a5=random.randrange(1, 4, 1)
 return [a0,a1,a2,a3,a4,a5]

# Part IV.b: function that implements the code solution
def algorithm(A):
 a=-1;b=-2;c=A[2];d=A[3]
 mySimulation = "a0=%d  a1=%d\n" % (A[0],A[1])
 mySimulation += "row  a  b  c d\n"
 mySimulation += "   1   %d %d %d %d\n" % (a,b,c,d)
 mySimulation += "   2   %d %d %d %d\n" % (a,b,c,d)
 a=A[0]
 mySimulation += "   2   %d %d %d %d\n" % (a,b,c,d)
 b=A[1]
 mySimulation += "   3   %d %d %d %d\n" % (a,b,c,d)
 while d>0:
   d-=1
   mySimulation += "   3   %d %d %d %d\n" % (a,b,c,d)
   if b<a:
     a-=A[4]
     mySimulation += "   7   %d %d %d %d\n" % (a,b,c,d)
     mySimulation += "   8                          111\n"
   if b>a:
     mySimulation += "   11                         222\n"
   else:
     b+=A[5]
     mySimulation += "   13  %d %d %d %d\n" % (a,b,c,d)
     mySimulation += "   14                     333\n"
 mySimulation += str(len(mySimulation))
 return str(mySimulation)

# Part IV.c: choose A with equivalent numbers of iterations
# for test, remove "#" bellow
while True:
 A=myRandom()
 mySimulation = algorithm(A)
 if 250<len(mySimulation)<300:
   #print (A)
   #print (mySimulation)
   break
]]
```

Contents

- Motivation
- Materials and methods
  - Questions to Discussion
    - Assessment with manual correction
    - **Assessment with code submission on the computer**
  - Proposal to integrate MCTest and VPL – applied in 2019
- Preliminary results and discussions
- Conclusion and Future Works

Method: Create question, **code submission on the computer**

#1239 1. Create 2 vectors $E1$ and $E2$ of integers with 22 positions each.
Read 22 elements by storing them in the $E1$ vector.
Fill in the vector $E2$ from $E1$ based on the following rule, while k is the index variable that will be used to access both vectors:

- if $k = 0$, $E2[k]$ receives smaller elements into $\{E1[21], E1[0], E1[1]\}$;
- if $k = 21$, $E2[k]$ receives smaller elements into $\{E1[20], E1[21], E1[0]\}$;
- if k its between 1 anb 21, that is, $1 \leq k < 21$, $E2[k]$ receivessmaller elements into $\{E1[k-1], E1[k], E1[k+1]\}$.

ATTENTION:
Submit the file **exam.java** (with the answer) and the file **model.txt**, containing only the text **Model: F**. First upload the java file and then the txt file.

Example:

```
input : 1 8 6 0 8 6 4 8 7 4 3 8 7 6 4 2 2 7 3 8 3 0
output: 0 1 0 0 0 4 4 4 4 3 3 3 6 4 2 2 2 2 3 3 0 0
```

Method: Create question, **code submission on the computer**

% Part I: description of the question

Create 2 vectors $ [[code:var1]] 1$ and $ [[code:var1]] 2$ of integers with $[[code:a0]]$ positions each.

Read $[[code:a0]]$ elements by storing them in the $ [[code:var1]] 1$ vector.

Fill in the vector $[[code:var1]] 2$ from $ [[code:var1]] 1 $ based on the following rule, while $ [[code:var0]] $ is the index variable that will be used to access both vectors:

```
\begin{itemize}
\item if $ [[code:var0]] = 0$,
$[[code:var1]] 2[ [[code:var0]] ]$  receives [[code:a1]] elements into  $\{[[code:var1]] 1[ [[code:a0-1]] ], [[code:var1]] 1[0], [[code:var1]] 1[1]\}$;

\item if $ [[code:var0]] = [[code:a0-1]]$,
$[[code:var1]] 2[ [[code:var0]] ]$ receives [[code:a1]] elements into  $\{[[code:var1]] 1[ [[code:a0-2]] ], [[code:var1]] 1[ [[code:a0-1]] ], [[code:var1]] 1[0]\}$;

\item if $ [[code:var0]]$ it's between $1$ anb $[[code:a0-1]]$, that is, $1\leq [[code:var0]] < [[code:a0-1]]$,
$ [[code:var1]] 2[ [[code:var0]] ]$ receives[[code:a1]] elements into  $\{ [[code:var1]] 1[ [[code:var0]] -1], [[code:var1]] 1[ [[code:var0]] ], [[code:var1]] 1[ [[code:var0]] +1]\}$.
\end{itemize}

\noindent\textbf{ATTENTION:}  \\  Submit the file \textbf{exam.java}  (with the answer) and the file \textbf{model.txt}, containing only the text \textbf{Model: [[code:model]]}. First upload the java file and then the txt file.
```

% Part II: the correct answer, like an example

```
\vspace{5mm}\noindent\textbf{Exemplo:}
\begin{verbatim}
[[code:mySimulation]]
\end{verbatim}
```

% Part III: python code between "[[def:" and "]]"

[[def:

```
import random, numpy as np

# Part III.a: create random variables
a_tam   = 2 # random.randrange(0,3,1)
a_inicio = 20
a0=random.randrange(a_inicio, a_inicio+a_tam+1, 1)
oper = ["larger","smaller"]
a1=random.choice(oper)
letters = ["A","B","C","D","E","F","G"]
model = letters[oper.index(a1)*(1+a_tam)+(a0-a_inicio)]
var0 = random.choice(["i","j","x","y","w","k","p"])
var1 = random.choice(letters)
global mySimulation
v1 = np.random.randint(9, size=a0)
v2 = np.zeros(a0, dtype='int')

# Part III.b: implements the code solution
mySimulation='input : '+' '.join([str(i) for i in v1])+'\n'
for i in range(a0):
 aux = [v1[(i-1)%a0], v1[i], v1[(i+1)%a0]]
 if a1==oper[0]: #max
   v2[i]= np.max(aux)
 if a1==oper[1]: #min
   v2[i]= np.min(aux)
mySimulation += 'output: ' + ' '.join([str(i) for i in v2])
```

]]

Contents

- Motivation
- Materials and methods
  - Questions to Discussion
    - Assessment with manual correction
    - Assessment with code submission on the computer
  - **Proposal to integrate MCTest and VPL – applied in 2019**
- Preliminary results and discussions
- Conclusion and Future Works

Proposal to integrate MCTest and VPL – applied in 2019

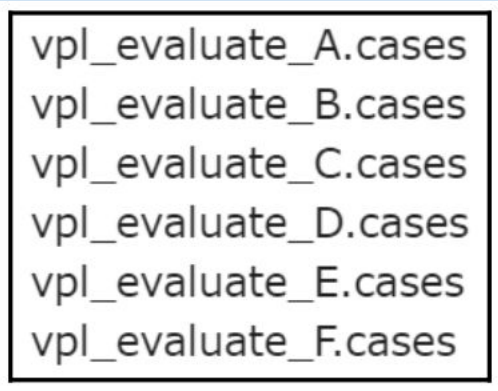
- Our main contribution: **Proposal to integrate MCTest and VPL**
  - MCTest:
    - Zampirolli, F., Teubl, F., and Batista, V. (2019). Online generator and corrector of parametric questions in hard copy useful for the elaboration of thousands of individualized exams. In CSEDU (1), pages 352–359.
  - Virtual Programming Lab (VPL):
    - Rodríguez-del Pino, J. C., Rubio Royo, E., and Hernández Figueroa, Z. (2012). A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. Conference: International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government
    - https://vpl.dis.ulpgc.es/

Proposal to integrate MCTest and VPL – applied in 2019

- Introductory Programming (IP) Course
- Course Blended-Learning (**IP-BL**) approach
- Exams
  - Manual (Exam1)
  - Computer (Project, Exam2 and Recovery Exam)
- 3 parametric questions each
- Six variations of each questions
- MCTest choose classrooms and questions and create a PDF file for each student

Proposal to integrate MCTest and VPL – applied in 2019

- After a Moodle VPL activity is created in runtime files, the teacher would have to add six text cases:



- and other files available at

  github.com/fzampirolli/mctest/VPL_modification/V1-select_using_second_file

Contents

Preliminary results and discussions

- Introductory Programming (IP) Course in 2019.1
- 1437 students in IP in 46 laboratory or IP-BL
- 8/46=17% Python; 36/46=78% Java
- 2/46=4% **IP-BL** applied a pseudocode-to-java (Portugol Studio tool):
  - 18 Multiple-Choice Tests (5%)
  - 17 Programming Exercise (5%)
  - Group project of 3 students (15%)
  - Exam1 (35%)
  - Exam2 (40%)
  - Class A1, 116 students (Santo André) – 86.21% approval
  - Class A2, 55 students (São Bernardo) – 54.55% approval

Preliminary results and discussions

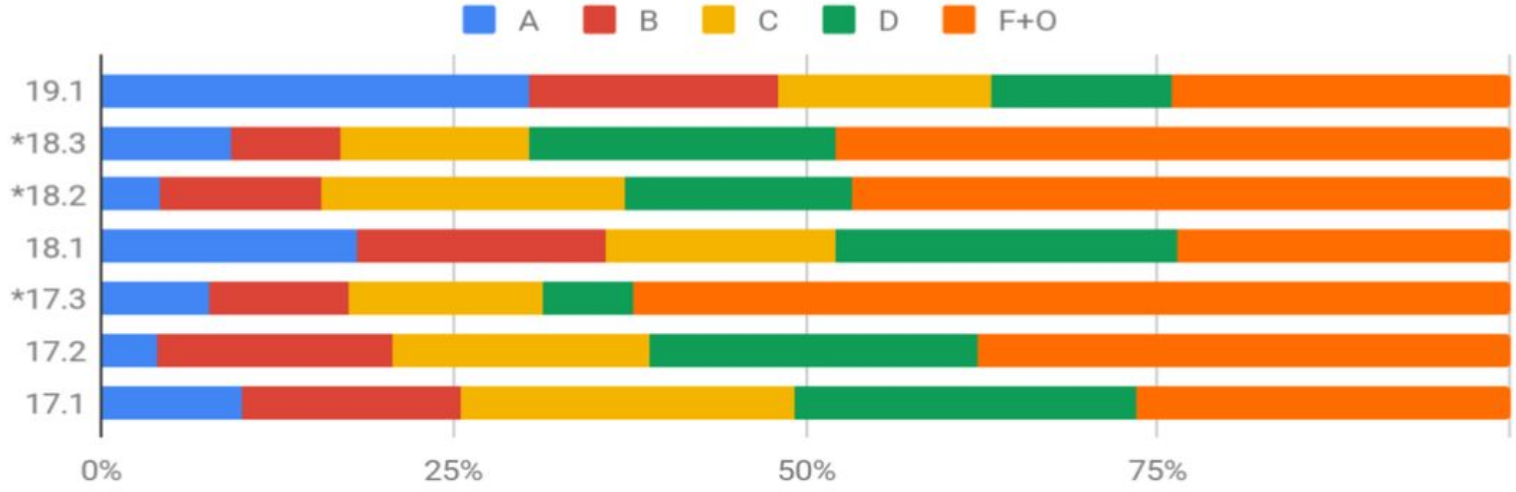


Figure 7. IP–BL performance in recent years. A high failure rate is observed in periods 18.3, 18.2, and 17.3, when an Exam2 = "F" rule was used, so that the student was required to take the recovery exam. The grade "O" means failure due to a high number of absences in a class (Source: The authors).

Future Work

- We have worked on an improvement, which is able to automatically choose the model/version based on the student name in Moodle/VPL. Consequently, the student would not need to submit a file informing the question model/version.

Thanks!

Questions?

{fzampirolli, paulo.pisani, marcelo.josko, guiou.kobayashi, francisco.fraga, denise.goya}@ufabc.edu.br, and heitor.rodrigues@aluno.ufabc.edu.br