Principle and Computer Simulation Model of Variation of Delastell’s cipher BIFID

M. Musilek and S. Hubalovsky

***Abstract*—**An interesting possibility to develop system programmer thinking of students of computer science is integration digital technology to non-trivial pencil and paper cipher system. The computer support gives us the opportunity of experimenting and creative modifications of the original idea. The paper describe introducing method of system approach, modeling and computer simulation to learning of algorithm development and programming for student of Computer Support of Archives specialization. The approach is based on creation of simulation program for encryption and decryption different types of ciphers. The paper describes the principles of polygraphic Delastell’s cipher BIFID and its variation as well as possibilities of encryption and decryption of the cipher using the computer simulation program.

***Keywords*—**Algorithmic thinking, Delastell’s cipher, education, historical encryption, programming.

I. Introduction

THE ability to create mathematical model and transform it to algorithm as well as to computer simulation program develops system thinking, skills and imagination. Regarding this fact the courses of algorithm development and programming are an inseparable part of study skills of students specializing in “Informatics” at high schools and secondary schools [1].

Learning of algorithm development and programming was/is often explained by the mathematical tasks, which can be clearly described, defined and developed by algorithm. Altogether, the exercises are based on rewriting the mathematical equations and formulas using algorithms and practicing the standard algorithm. The complexity and integration of system approach to learning of algorithm development and programming is missing [2]. Students, who do not have sufficient mathematical experience, do not understand algorithm as well as programming task. In such type of learning the students cannot see the context with problems that occur in real life. Learning of algorithm development escapes them, and the result is indifference or resistance to the algorithm and subsequent programming.

Rather than rewriting the mathematical task in the learning of programming the new method based on introducing the system approach, modeling and simulation is used in learning of students of *Computer Support of Archives* specialization at Faculty of Art, University of Hradec Kralove (see e.g. [3] – [7]).

The mentioned approach is demonstrated by case study of using of polygraphic Delastell’s cipher BIFID and its variation. The computer simulation of the case studies is realized and visualized in Java Script programming language.

II. Theoretical Background

A. Principles of Polygraphic Substitution Cipher BIFID

Polygraphic substitution cipher BIFID [8] combines fractionation of substitution tables with transposition. The result is polygraphic substitution cipher. The specified cipher operations are performed with a group of five digits in the basic variant, i.e. the substitution is clearly intended for a group of five symbols by used encryption table (generally called Polybius square) and by agreed manner of transposition of given numerical mid-text. Below example will clarify the situation.

The following message has to be encrypt by polygraphic ciphers BIFID: "*Both men are employed on the Faculty of Science*." The message has to be first rewritten to five-letter’s groups, ignoring the spaces between the words and the last group is complete to five characters:

```
BOTHM   ENARE   EMPLO   YEDON
FACUL   TYOFS   CIENC   EKLMN
```

Encryption table (Polybius square) will be input based on passwords University of Hradec Kralove, where letters I and J are connected (as is common in English) to one field of square – see Table 1:

Table 1 Encryption table

	1	2	3	4	5
1	U	N	I/J	V	E
2	R	S	T	Y	O
3	F	H	A	D	C

Stepan Hubalovsky is assoc. prof. at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, stepan.hubalovsky@uhk.cz.

Michal Musilek is assistant professor at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, michal.musilek@uhk.cz.

A	B	C	D	E	F	G
382	104	132	248	646	114	97
H	**I**	**J**	**K**	**L**	**M**	**N**
324	323	0	32	194	99	326
O	**P**	**Q**	**R**	**S**	**T**	**U**
370	106	3	300	302	432	130
V	**W**	**X**	**Y**	**Z**		
56	114	11	72	2		

4	K	L	B	G	M
5	P	Q	W	X	Z

First, the text will be encrypt into to numeric mid-text:

```
BOTHM  ENARE  EMPLO  YEDON
42234  11321  14542  21321
35325  52315  55125  45452

FACUL  TYOFS  CIENC  EKLMN
33314  22232  31113  14441
13512  34512  53525  51252
```

The cipher text is reached from the mid-text in the second phase of encryption. The principle of the creation of the cipher from the mid-text is as follows:

Twice two letters are taken from the first row of five-digit group;

Then last letter of the first row is connected with the first letters of the second row;

Finally twice two letters are taken from the second row.

For the second phase of the encryption the same table as in the first phase will be used:

```
42234  11321  14542  21321
35325  52315  55125  45452
LTKWO  UHETE  VXOPO  RHVXQ

33314  22232  31113  14441
13512  34512  53525  51252
AFKCN  STTMN  FUCCO  VGENQ
```

It is clear from the example that this type of encryption is more complicated than the encryption of other substitution ciphers, (simple substitution, bigram substitution of type Playfair cipher or Four-square cipher).

The complexity of the BIFID cipher can be proved by increasing of information entropy of the ciphertext, e.g. if this cipher is generally worse decipherable.

B. *Index of Coincidence and Information Entropy*

The text of the first chapter of the novel Oliwer Twist by Charles Dickens has been chosen to calculate the information entropy. Table 2 shows frequency analysis of the plain text.

Table 2 Frequency of characters in plain text

The effectiveness of encryption algorithm may be calculated based on index of coincidence. The index of coincidence was introduced to cryptanalysis by William Frederick Friedman [9] (1891-1969). If the frequencies of individual letters of the alphabet is n_i, the number of different characters forming the alphabet k and total number of characters of the analyzed text N, then the index of coincidence define IC is given by formula (1):

$$IC = \sum_{i=1}^{k} \frac{n_i \cdot (n_i - 1)}{N \cdot (N - 1)} \tag{1}$$

The approximate formula is used in cryptoanalysis that gives a more accurate value of the IC, for long analyzed text. We have also used the following approximate formula, because we analyzed the texts of length of the thousands of characters. The value of p_i is the relative frequency (posteriori probability) of occurrence of the i^{th} character of alphabet:

$$IC = \sum_{i=1}^{k} p_i^2 \tag{2}$$

Another variable that we can be used for measurement of the effectiveness of an encryption algorithm informatics entropy is so called Shannon entropy Claude Elwood Shannon (1916-2001)). The index of coincidence of the information entropy H is defined as follows:

$$H = -\sum_{i=1}^{k} p_i \cdot \log_2 p_i \tag{3}$$

Where p_i is again the relative frequency of occurrence of the i^{th} character of alphabet.

The index of coincidence for given plaintext (Oliver Twist by Charles Dickens) is IC = 0.0657, informatics entropy text is H = 4.16.

After application of the above procedure to BIFID cipher and to Polybius square obtained by using password University of Hradec Kralove, the frequency of the character is shown in Table 3.

Table 3 Frequency of characters in cipher text

A	B	C	D	E	F	G
316	129	179	125	189	246	50
H	**I**	**J**	**K**	**L**	**M**	**N**
427	288	0	111	109	83	326
O	**P**	**Q**	**R**	**S**	**T**	**U**
269	184	137	238	290	392	223
V	**W**	**X**	**Y**	**Z**		
135	137	71	147	118		

The index of coincidence of the ciphertext is IC = 0.0500, text informatics entropy increases to H = 4.47. For comparison, the plaintext was encrypt by bigram substitution cipher Playfair using the same Polybiova squares. The resulted ciphertext has length of 5080 characters whose extension was due to completion of double consonants. The index of coincidence is H = 0.0524 and informatics entropy text is H = 4.39, which is consistent with the expectation that bigram substitution will have for the same plaintext higher index of coincidence and lower informatics entropy than polygrams substitution.

Ideally random text, which uses 25 English letters (the letter J is not present even in our chosen plaintext, nor in any of the cipher text), is minimal coincidence index IC = 1/25 = 0.0400 and informatics maximum entropy $H = \log_2 25 = 4.64$.

Let us summarize the values of the statistical characteristics of the four previously mentioned texts in Table 4.

Table 4 Values of coincidence index *IC* and informatics entropy *H*.

	Plain text	Cipher text Playfair	Cipher text BIFID	Ideally random text
IC	0.0657	0.0524	0.0500	0.0400
H	4.16	4.39	4.47	4.64

C. Modification of Delastell's Cipher BIFID

Delastell's cipher BIFID can be varied by number of different ways. The first variation changes the dimensions of Polybius square from 5 x 5 cells to 6 x 6. Table of 6 x 6 characters can be used for encryption of the 26 letters of the English alphabet and 10 digits. Larger tables enable encryption of larger alphabets. For example Cyrillic has 33 characters, which requires the use of a square with 36 cells. The remaining three cells have to be supplemented by other appropriate symbols, e.g. exclamation mark, question mark and a plus sign. Czech alphabet with accents consists of a total of 42 different graphemes. If the length of vowels is not distinguish, remains 35 different characters and last cell can be supplemented by e.g. exclamation point.

Samples of encryption tables - Polybius squares - for the Russian and Czech alphabet are shown on the Table 5 and Table 6.

Table 5 Encryption table for Russian alphabet

	1	2	3	4	5	6
1	А	Л	Е	К	С	Н
2	Д	Р	В	И	Й	!
3	Б	Г	Ё	Ж	З	М
4	О	П	Т	У	Ф	?
5	Х	Ц	Ч	Ш	Щ	+
6	Ъ	Ы	Ь	Э	Ю	Я

Table 6 Encryption table for Czech alphabet

	1	2	3	4	5	6
1	A	L	E	X	N	D
2	R	V	I	K	Y	!
3	B	C	Č	Ď	F	G
4	H	J	K	L	M	Ň
5	O	P	Q	Ř	S	Š
6	T	Ť	U	V	W	Z

For both tables was chosen password "Александр Великий", respectively "Alexander Veliký“.

Other variations of Polybius cipher are based on permutations of used transposition. The letters can be associated not only horizontally but also upward and downward – see [*]. The message “Both men are employed on the Faculty of Science." Is in this case encrypt as follows:

- The first phase is in all three cases the same.
- The second phase is as follows for horizontal association - read horizontally in the first group 42, 23, 41, 53, 25; in the second group 11, 32, 15, 23, 15, etc.:

```
42234  11321  14542  21321  →
15325  52315  55125  45452  →
LTKWO  UHETE  VXOPO  RHVXQ

33314  22232  31113  14441  →
13512  34512  53525  51252  →
AFKCN  STTMN  FUCCO  VGENQ
```

- For association obliquely upwards - read obliquely upward in the first group 12, 52, 33, 24, 54; in the second 51, 23, 32, 11, 51, etc.:

```
42234  11321  14542  21321  ↗
15325  52315  55125  45452  ↗
NQAYX  PTHUP  XZVSP  KWLPS

33314  22232  31113  14441  ↗
13512  34512  53525  51252  ↗
IAPVT  HLWNS  PFPTW  XVYPR
```

- Finally, association obliquely downwards - read obliquely downward in the first group 45, 23, 22, 35, 41; in the second 12, 13, 31, 25, 15, etc.:

```
42234  11321  14542  21321  ↘
15325  52315  55125  45452  ↘
GTSCK  NIFOE  EKQMO  OVCSV

33314  22232  31113  14441  ↘
```

```
13512  34512  53525  51252  ↘
ACFNK  YORHT  AENEC  ULMLE
```

Another way haw to complicate work to codebreakers is to use two encryption tables. This alternative example is shown on the example - see Table 7. To convert the letters into pairs of digits the Polybius square with password "University of Hradec Kralove" is used. The retransfer of pairs of numbers to Polybius square the password "Thales of Miletus" is used – see Table 8.

Table 7 Direct encryption table

	1	2	3	4	5
1	U	N	I/J	V	E
2	R	S	T	Y	O
3	F	H	A	D	C
4	K	L	B	G	M
5	P	Q	W	X	Z

Table 8 Reverse encryption table

	1	2	3	4	5
1	T	H	A	L	E
2	S	O	F	M	I
3	U	B	C	D	G
4	K	N	P	Q	R
5	V	W	X	Y	Z

```
BOTHM  ENARE  EMPLO  YEDON
42234  11321  14542  21321
15325  52315  55125  45452
NFKXI  TBEFE  LYIVI  SBLYW

FACUL  TYOFS  CIENC  EKLMN
33314  22232  31113  14441
13512  34512  53525  51252
CUKGH  OFFRH  UTGGI  LQEHW
```

Interesting variation is to encrypt text instead of group of five letter by words. After “words” encryption the text is redistributed to five-groups. Leave text in groups corresponding to the lengths of individual words would be too much guidance, the codebreakers would have the work very easy. They would know not only the length of all words in the text, but also the same word would always replace the same group of letters. After the division into five-letters groups, there is no additional information on the length of words in the text. We get the ciphertext index of coincidence *IC* = 0.0511 and informatics entropy *H* = 4.45. The strength of the cipher is therefore somewhere between five-group BIFID cipher and Playfair cipher.

The challenge for Informatics is automatic decrypting such ciphers. Regarding the fact the length of each word is unknown, the word of different length has to be try and in vocabulary has to be used to check whether the group voice makes sense.

Finally, the most complicated method is to encrypt by groups with random length, e.g. a length of a finite set of odd natural numbers {3, 5, 7, 9, 11, 13}. Such a sequence can be easily produced by normal dice, which generates integers from the set {1, 2, 3, 4, 5, 6}. The odd sequence of numbers is reached from the dice number by multiplication by two and add by one. In this case the creation of algorithm that efficiently decrypt the ciphertext to the original message - plain text by using a dictionary of a given language is non-trivial task, very suitable for the application system approach.

Recently described cryptosystem is very interesting. From the code breaker it is a classic pencil and paper cipher system, supplemented only by dice. The decoders has to use for decryption a special computer program with dictionary databases containing words of the language appropriately modified (e.g. the Czech language with the removal of diacritics) and sorted according to the length.

III. Computer Simulation Program

The web page for encryption and decryption of the Delastell’s cipher BIFID is created in JavaScript. Each function realized the separate task. The algorithm of the program is clear directly from the program code based on algorithm.

A. Description of program code

The first part of the program code is declaration of the variables. Generally they are arrays of different lengths with names `square`, `used`, `writecharacter`, `top`, `bottom`:

```
<script type="text/javascript"
language="JavaScript">
<!--
var square = new Array(25);
var used = new Array(27);
var writecharacter= new Array(27);
var top = new Array(50);
var bottom = new Array(50);
```

Array `writecharacter` is responsible for conversion of the number to alphabet character:

```
writecharacter =
"*,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S
,T,U,V,W,X,Y,Z".split(",");
```

Backward conversion is given by function `order`:

```
function order(character)
{
var p;
if (character == "A") {p = 1};
if (character == "B") {p = 2};
if (character == "C") {p = 3};
if (character == "D") {p = 4};
if (character == "E") {p = 5};
if (character == "F") {p = 6};
if (character == "G") {p = 7};
if (character == "H") {p = 8};
if (character == "I") {p = 9};
if (character == "J") {p = 10};
if (character == "K") {p = 11};
if (character == "L") {p = 12};
if (character == "M") {p = 13};
if (character == "N") {p = 14};
if (character == "O") {p = 15};
if (character == "P") {p = 16};
if (character == "Q") {p = 17};
if (character == "R") {p = 18};
if (character == "S") {p = 19};
if (character == "T") {p = 20};
if (character == "U") {p = 21};
if (character == "V") {p = 22};
if (character == "W") {p = 23};
if (character == "X") {p = 24};
if (character == "Y") {p = 25};
if (character == "Z") {p = 26};
return(p);
}
```

The procedure order() is used not only for cipher BIFID, but also for many other substitution ciphers. The same is also for function preppasword, that all letters in the text string converted to uppercase, remove accents, omitting any spaces and punctuation, and also removes duplicates. To remove duplicates the nested loop is used:

```
for (j = 0; j < i; j++)
  {
     if (bezdk.charAt(i) ==
bezdk.charAt(j)) {add = 0};
  }
```

Spaces and punctuation are removed by setting the value of the variable add to zero. The password is created only by characters which has value of variable add equal to one:

```
function preppassword()
{
pswrd = document.formular.heslo.value;
paswd = pswrd.toUpperCase();
bezdk = "";
noveh = "";
n = paswd.length;
for (i = 0; i < n; i++)
{
character = paswd.charAt(i);
if (character == "Á") {character = "A"};
if (character == "É") {character = "E"};
if (character == "Ě") {character = "E"};
if (character == "Í") {character = "I"};
if (character == "Ó") {character = "O"};
if (character == "Ú") {character = "U"};
if (character == "Ů") {character = "U"};
if (character == "Ý") {character = "Y"};
if (character == "Č") {character = "C"};
if (character == "Ď") {character = "D"};
if (character == "Ň") {character = "N"};
if (character == "Ř") {character = "R"};
if (character == "Š") {character = "S"};
if (character == "Ť") {character = "T"};
if (character == "Ž") {character = "Z"};
bezdk += character;
pridat = 1;
for (j = 0; j < i; j++)
{
if (bezdk.charAt(i) == bezdk.charAt(j))
{pridat = 0};
}
if (bezdk.charAt(i) == " "){pridat = 0};
if (bezdk.charAt(i) == "."){pridat = 0};
if (bezdk.charAt(i) == ","){pridat = 0};
if (bezdk.charAt(i) == ";"){pridat = 0};
if (bezdk.charAt(i) == "!"){pridat = 0};
if (bezdk.charAt(i) == "?"){pridat = 0};
if (pridat == 1) {noveh += character};
}
document.formular.heslo.value = noveh;
}
```

The following procedure create can be used not only for cipher BIFID, but also for all ciphers based on Polybius square. This function fills the cells of table 5 x 5 characters by letters of passwords and when these letters are finished the function fills the remaining letters of the alphabet. Because the scripting language cannot use multi-dimensional arrays only one-dimensional array is used. The index of element in the square table corresponds to the sum of quintuple of the serial number of the table row (rows are numbered from 0 to 4) and the serial number of the column (are numbered from 0 to 4). Letters in Polybius square are thus saved in the one-dimensional array with indexes from 0 to 24:

```
function create()
{
preppassword();
pswrd = document.form.password.value;
paswd = pswrd.toUpperCase();
n = paswd.length;
for (j = 1; j < 27; j++) {used[j] = 0};
if (document.form.skip[0].checked)
    { used [10] = 1};
if (document.form.skip[1].checked)
    { used [17] = 1};
if (document.form.skip[2].checked)
    { used [23] = 1};
if (document.form.skip[3].checked)
    { used [26] = 1};
for (i = 0; i < n; i++)
{
square[i] = paswd.charAt(i);
j = order(square[i]);
used[j] = 1;
};
for (i = n; i < 25; i++)
{
other = 1;
for (j = 1; j < 27; j++)
{
if ((used[j] == 0) && (other == 1))
{
square[i] = writechar[j];
used[j] = 1;
other = 0;
}
}
}
}
```

The following procedures `show` shows cipher square called by method `alert`:

```
function show()
{
create();
output = square[0] + " " + square[1] +
  " " + square[2];
output += " " + square[3] + " " +
  square[4];
output += "\n" + square[5] + " " +
  square[6] + " "
output += square[7] + " " + square[8] +
  " " + square[9];
output += "\n" + square[10] + " " +
  square[11] + " ";
output += square[12] + " " + square[13]
  + " ";
output += square[14] + "\n" + square[15]
  + " ";
output += square[16] + " " + square[17]
  + " ";
output += square[18] + " " + square[19];
  + "\n";
output += square[20] + " " + square[21]
  + " ";
output += square[22] + " " + square[23]
  + " ";
output += square[24];
alert(output);
}
```

Help procedure `clear` sets value of all elements of temporary arrays `top` and bottom to `zero`:

```
function clear()
{
for (i = 0; i < 50; i++)
{
top[i] = 0;
bottom [i] = 0;
}
}
```

Main program `action` calls above procedures a realized all steps needed to ciphering and deciphering:

```
function action()
{
create();
clear ();
document.form.topnumber.value = '';
document.form.bottomnumber.value = '';
otxt = document.form.plaintext.value + '
';
ot = otxt.toUpperCase();
n  = ot.length;
j = 0;
codetext = "";
for (i = 0; i < n; i++)
{
character = ot.charAt(i);
if (character == "Á") {character = "A"};
if (character == "É") {character = "E"};
if (character == "Ě") {character = "E"};
if (character == "Í") {character = "I"};
if (character == "Ó") {character = "O"};
if (character == "Ú") {character = "U"};
if (character == "Ů") {character = "U"};
if (character == "Ý") {character = "Y"};
if (character == "Č") {character = "C"};
if (character == "Ď") {character = "D"};
if (character == "Ň") {character = "N"};
if (character == "Ř") {character = "R"};
if (character == "Š") {character = "S"};
if (character == "Ť") {character = "T"};
if (character == "Ž") {character = "Z"};
```

Important separation character of plaintext is the spacebar – encryption of character array is run between the previous and current space:

```
if (character == " ")
{
for (k = 0; k < j; k++)
{
if (k % 2 == 0)
{
l = k + 1;
if (l >= j)
{
codetext += square[5*(top[k]-
1)+1*(bottom[0]-1)];
}
else
{
codetext += square[5*(top [k]-
1)+1*(top[l]-1)];
};
};
}
if (j % 2 == 0) {k = 0} else {k = 1};
while (k < j)
{
codetext += square[5*(bottom[k]-
1)+1*(bottom[k+1]-1)];
k += 2;
}
codetext += " ";
document.form.codetext.value = codetext;
document.form.topnumber.value += '';
document.form.bottomnumber.value += '';
j = 0;
}
else
{
for (k = 0; k < 25; k++)
{
if (square[k] == character)
{
cod = k;
top [j] = Math.floor(kod / 5) + 1;
bottom[j] = (kod % 5) + 1;
document.form.topnumber.value += top
[j];
document.form.bottomnumber.value +=
bottom[j];
j++;
};
}
};
}
document.form.codetext.value = codetext;
}
```

Similarly, the procedure `back` realized decryption. The procedure is similar to the procedure for encryption, but works with other pairs of characters of inter-text composed of digits indicating the serial numbers of rows of columns of encryption key (i.e. password of Polybius square):

```
function back()
{
create();
clear();
document.form.topnumber.value = '';
document.form.bottomnumber.value = '';
otxt = document.form.plaintext.value + '
';
ot = otxt.toUpperCase();
n= ot.length;
j = 0;
codetext = "";
for (i = 0; i < n; i++)
{
character = ot.charAt(i);
if (character == " ")
{
for (k = 0; k < (j/2); k++)
{
bottom[k] = top[j/2+k];
top[j/2+k] = 0;
}
for (k = 0; k < (j/2); k++)
{
codetext += square[5*(top[k]-
1)+1*(bottom[k]-1)];
document.form.topnumber.value += top[k];
document.form.bottomnumber.value +=
bottom[k];
}
codetext += " ";
document.form.codetext.value = cedetext;
document.form.topnumber.value += '';
document.form.bottomnumber.value += '';
j = 0;
}
else
{
for (k = 0; k < 25; k++)
{
if (square[k] == character)
{
cod = k;
top[j] = Math.floor(kod / 5) + 1;
top[j+1] = (cod % 5) + 1;
j += 2;
};
}
};
}
document.form.codetext.value = codetext;
}
```

B. *Error correction*

Based on cipher table (Table 9) the letter B has coordinate 4 and 3, red number should not be 1, it should be 3.

Table 9 Cipher table

	1	2	3	4	5
1	U	N	I/J	V	E
2	R	S	T	Y	O
3	F	H	A	D	C
4	K	L	B	G	M
5	P	Q	W	X	Z

```
BOTHM   ENARE   EMPLO   YEDON
42234   11321   14542   21321
15325   52315   55125   45452

FACUL   TYOFS   CIENC   EKLMN
33314   22232   31113   14441
13512   34512   53525   51252
```

The third letter of cipher text change from K to B:

```
42234   11321   14542   21321
35325   52315   55125   45452
LTBWO   UHETE   VXOPO   RHVXQ

33314   22232   31113   14441
13512   34512   53525   51252
AFKCN   STTMN   FUCCO   VGENQ
```

C. *Visualization of the simulation model*

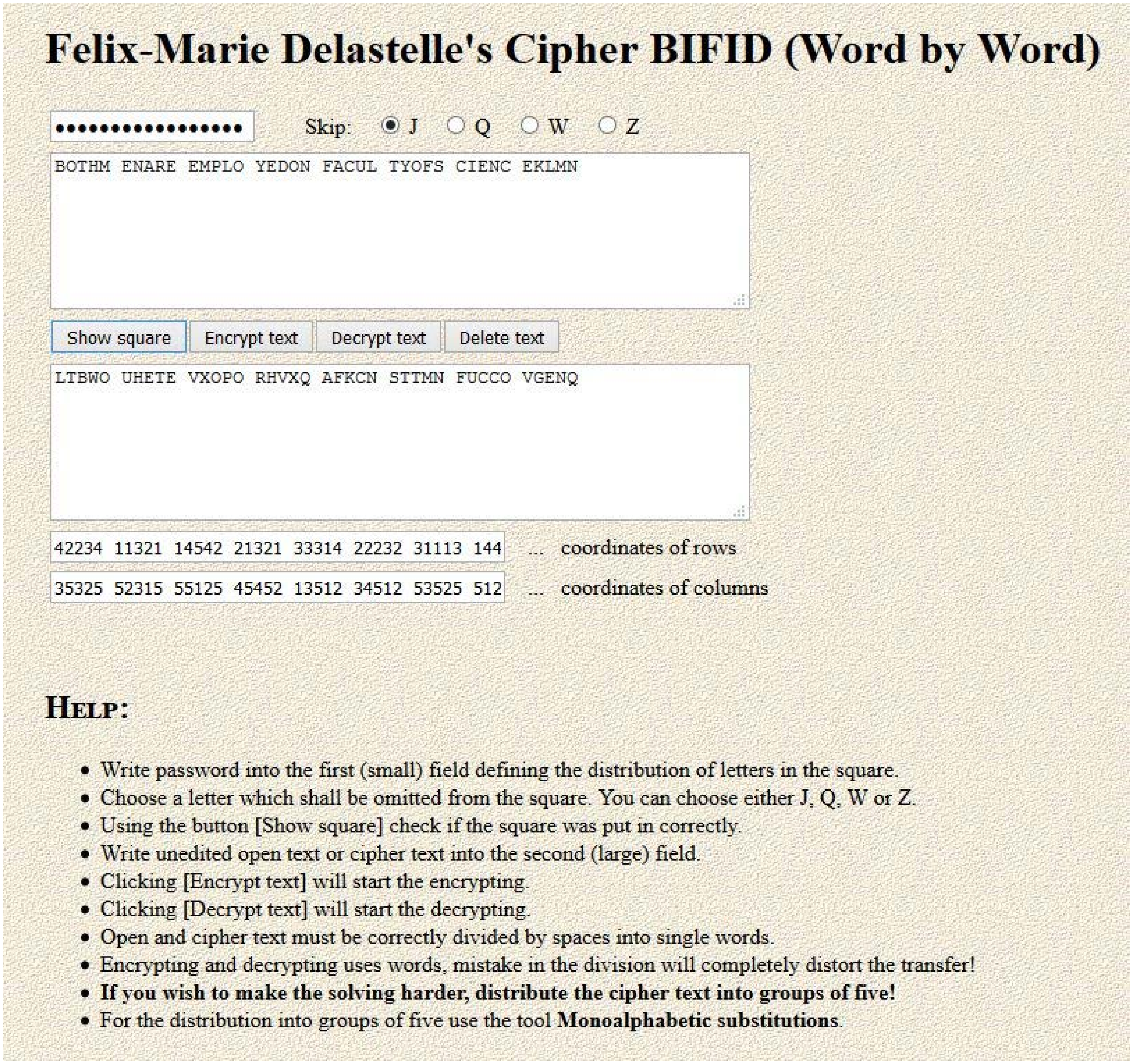
The web page shown on the Figure 1 represents visualization of the simulation model.

IV. Conclusion

Case study demonstrated in the paper highlights system approach, modeling and simulation to learning of programming of students of humanities. Students of humanities are not able to learn algorithm development and programming standardly based on mathematical task. The presented approach is based on introducing of encryption issues, which increase motivation of students of humanities and develop their algorithmic thinking and skill for algorithms creation.

Acknowledgment

This research has been supported by: Specific research project Specific research project of University of Hradec Kralove, Faculty of Science in 2015 No. 2108.



Felix-Marie Delastelle's Cipher BIFID (Word by Word)

•••••••••••••••• Skip: ● J ○ Q ○ W ○ Z

```
BOTHM ENARE EMPLO YEDON FACUL TYOFS CIENC EKLMN
```

```
LTBWO UHETE VXOPO RHVXQ AFKCN STTMN FUCCO VGENQ
```

```
42234 11321 14542 21321 33314 22232 31113 144
```
... coordinates of rows

```
35325 52315 55125 45452 13512 34512 53525 512
```
... coordinates of columns

HELP:

- Write password into the first (small) field defining the distribution of letters in the square.
- Choose a letter which shall be omitted from the square. You can choose either J, Q, W or Z.
- Using the button [Show square] check if the square was put in correctly.
- Write unedited open text or cipher text into the second (large) field.
- Clicking [Encrypt text] will start the encrypting.
- Clicking [Decrypt text] will start the decrypting.
- Open and cipher text must be correctly divided by spaces into single words.
- Encrypting and decrypting uses words, mistake in the division will completely distort the transfer!
- **If you wish to make the solving harder, distribute the cipher text into groups of five!**
- For the distribution into groups of five use the tool **Monoalphabetic substitutions**.

Figure 1 The web page for encryption and decryption of the Delastell's cipher BIFID

REFERENCES

[1] S. Hubalovsky, M. Musílek, "*Automatic cryptoanalysis of the monoalphabetical substitution as a method of the system approach in the algorithm development thinking*". International journal of applied mathematics and informatics. vol. 4, No. 4, 2010.

[2] V. Jehlicka, "Interdisciplinary relations in teaching of programming", in *Proc. WSEAS/IASME Applied computing conference 2010 (ACC'10)*, WSEAS Press, Timisoara, Romania, 2010. pp 33-39.

[3] P. Hanzalová, Š. Hubálovský, M. Musílek, "Automatic cryptoanalysis of the short monoalphabetical substituted cipher text". *Visualization, imaging and simulation (VIS '12)*. WSEAS Press, 2012.

[4] S. Hartmann, "The World as a Process: Simulations in the Natural and Social Sciences", In R. Hegselmann, et al., *Modelling and Simulation in the Social Sciences from the Philosophy of Science Point of View*, Theory and Decision Library. Dordrecht: Kluwer, 1996, pp. 77–100.

[5] J. A. Sokolowski, C. M. Banks, "Principles of Modeling and Simulation – A Multidisciplinary Approach", *Wiley Publication*, New Jersey, 2009, pp. 121-141.

[6] J. Bailer, M. Daniela, "Tracing the Development of Models in the Philosophy of Science", *Magnani, Nersessian and Thagard*, 1999, pp. 23-40.

[7] S. Hubalovsky, M. Musílek, "Automatic cryptoanalysis of the monoalphabetical substitution as a method of the system approach in the algorithm development thinking", *International journal of applied mathematics and informatics*, Vol.4, No.4, 2010, pp. 92-102.

[8] American Cryptogram Associaton, "*The ACA and You – A handbook for the members of the American Cryptogram Association.*" ACA 2005

[9] D. Khan, "*The Codebreakers. The Story of Secret Writing.*" Scribner 1967.

Michal Musilek was born in 1963 in Czech Republic. He obtained master degree in education of mathematics and physics in 1988, in education of computer science in 1993 and doctor degree in theory of education in physics in 2009 all in Faculty of Education, University of Hradec Kralove, Czech Republic. He works as assistant professor on University of Hradec Kralove. His scientific activities are theory of education in informatics includes children’s programming languages, using ICT in education of mathematics and physics include computer modeling and simulation.

Stepan Hubalovsky was born in Trutnov, Czech Republic in 1970, he obtained master degree in education of mathematics, physics and computer science in 1995 and doctor degree in theory of education in physics in 1998 both in Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic. He worked 5 years as master of mathematics, physics and computer science on several secondary schools. He works as assistant professor on University of Hradec Kralove from 2006. He interested in algorithm development, programming, system approach, computer simulation and modelling. Assoc. prof. RNDr. Stepan Hubalovsky, Ph.D. is member of Union of Czech Mathematicians and Physicist.