

# Effective determining of the degree of similarity of selected properties of objects through characteristic text strings

Petr Voborník

**Abstract**—The article illustrates the new unique coding method for describing the properties of different objects of the virtual and real world into the so-called characteristic text strings. The basic description of the object can be summarized into a very short text string by this method. These strings are comprised only of the basic characters of the alphabet and numbers and it is easily to readable by humans. These strings can be easily and very quickly compared with each other and determine mutual difference of described objects. Total and partial difference (differences in specific properties) may be calculated at any time. This enables us to easily find the object that is most dissimilar from or most similar to the pattern object. The whole principle of characteristic text strings is demonstrated on generating random elements in the test questions. Thanks to this principle, a complicated structure can be described of each randomly generated question by the short string. On the basis of this, such procedures can be suggested for repeatedly used questions that ensure generating of the most different version from versions used previously. The applicability is thus of each of the test questions maximized. Another area of use of characteristic text strings are then presented, such as web online shops, vehicle license plate, genetic algorithms, and more.

**Keywords**—Characteristic text strings, eShop, random elements, test questions.

## I. INTRODUCTION

ONLINE testing system, the *Universal Testing Environment*, allows creating graphically rich, interactive and multimedia enhanced questions of all sorts [1]. In order really not to limit the creative potential of the authors of the test questions, a respective language, QML<sup>1</sup> (based on XML and XAML [2]), was created for defining the structure of the questions. In addition to text, the QML allows the use of vector and bitmap graphics of any kind, animations, and various types of random values at any part of the question. This is achieved by the use of random elements [3] for selecting version of the question, generating random numbers and characters, selecting the text string from multiple variants, and mixing of inner elements structure of the questions.

Like with the selection of questions used in a test [4], it is desirable in this case that the internal mixing is not always entirely random, but it rather provides the opportunity to

generate the greatest possible variation of each question that is repeatedly used with the same tested user. Thanks to this, the selection process of random elements in each question should prefer those variations, or their combinations, which would differ the most from the variations used previously.

Determining the degree of difference between individual combinations of random elements should not be computationally complex (e.g. such as multi-distance spatial cluster analysis, see [5]), it should be easily storable in a database, and the structure of such record should be extensible for possible future development of the question.

## II. PRINCIPLE OF COMPARISON OF STRINGS

Firstly, the selected method will be explained using an easily presentable example: comparing of the persons [6]. Table 1 contains five human properties which will be compared for each person and a description of their data types and ranges (metadata).

Table 1. Metadata of persons' properties for comparison

Property	Index	Type	Min	Max	Range	Units	Weight
Height	1	cardinal	40	220	181	cm	1.5
Weight	2	cardinal	1	256	256	kg	0.9
Age	3	cardinal	0	127	128	years	1.0
Sex	4	nominal	1	2	2	-	10.0
Color of eyes	5	nominal	1	6	6	-	2.0

The *range* parameter determines how many possible values the property can take on. The process of its calculation is shown in Equation 1.

$$range_i = max_i - min_i + 1 \quad (1)$$

Properties are differentiated into two main types: *cardinal* and *nominal*. For the cardinal type, differences between values of individual subjects can be directly and precisely calculated, whereas for the nominal values it can only be determined

This research has been supported by Specific research project of University of Hradec Kralove, Faculty of Science in 2015.

P. Voborník is with the Department of Informatics, Faculty of Science, University of Hradec Králové, Rokitanského 62, Hradec Králové, 500 03, Czech Republic (e-mail: petr.vobornik@uhk.cz, orcid: 0000-0003-1841-3455).

<sup>1</sup> QML - Questions Markup Language [3]

whether the values are identical or not [7]. The calculation of the difference of these two models will therefore vary.

#### A. Cardinal values

Each compared subject can essentially be expressed as a vector (e.g.  $\vec{a}$  or  $\vec{b}$ ) comprised of the values of individual properties. If each of the cardinal properties had the same weight (e.g. if the difference of one year of age was as significant for the comparison as the difference of 1 cm of height or 1 kg of weight), the difference is in this case formed by a scalar product of a unit vector and a vector composed of the absolute differences of the individual components of the vectors  $\vec{a}$  and  $\vec{b}$  (see Equation 2).

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \begin{matrix} \text{is} \\ \text{different} \\ \text{from} \end{matrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} |a_1 - b_1| \\ |a_2 - b_2| \\ \vdots \\ |a_n - b_n| \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \quad (2)$$

Calculating differences of cardinal properties for entities A and B (Equation 2) can also be expressed using the sum of absolute values of differences of individual properties as shown in the Equation 3.

$$\sum_{i=1}^n |a_i - b_i| \quad (3)$$

If differences for each property had a different impact on the overall assessment of the overall similarity of two entities, their weights, which are part of the metadata, can then be taken into account in the calculation ( $\vec{v}$ , see last column in Table 1). Calculation of differences of the cardinal properties of entities A and B with the addition of weights is then shown in Equation 4<sup>2</sup>.

$$R_{car.} = \sum_{i=1}^n (|a_i - b_i| \cdot v_i) \quad (4)$$

#### B. Nominal values

For nominal values, although these categories are identified by numbers (for coding), it cannot be determined how much they differ, but only if they are identical or not. If both values are equal, the result is 0, if they are different the result is 1. In this case, either the comparison operator ( $\neq$ ) can be used for both nominal values, or the  $sgn^3$  function to the absolute value of their difference can be applied (see Equation 5). The degree of influence of both types of values on the total difference can be determined by setting the appropriate weight.

<sup>2</sup> vector of weights would then replace the unit vector in a vector terms (see Equation 2)

$$R_{nom.} = \sum_{i=1}^m (sgn|a_i - b_i| \cdot v_i) \quad (5)$$

#### C. Total difference

The total difference is made up of the sum of the differences of cardinal and nominal values (see Equation 6).

$$R = R_{car.} + R_{nom.} \quad (6)$$

The procedure for particular example of a comparison of two persons is shown in Table 2. Differences for individual types of values are calculated with the above equations that are selected according to the types of values (see col. *Type* in Table 1).

Table 2. Sample of procedure for calculating the difference between persons A and B

Property	Person A	Person B	Difference	Weight	Weighted difference
Height	175	165	10	1.5	15.0
Weight	85	60	25	0.9	22.5
Age	35	25	10	1.0	10.0
Sex	2	1	1	10.0	10.0
Color of eyes	5	1	1	2.0	2.0
<b>Total</b>			<b>47</b>		<b>59.5</b>

The resulting difference of persons A and B has a value of 59.5. If we considered an extra person C with the same values in all properties as with person B, with the only difference that its height would be 166 cm, then the difference between person A and C would have a value of 58. It follows that the person C would be more similar to person A than to person B. This way, the person that is most similar to or most dissimilar from the pattern<sup>4</sup> person A (from aspect of compared properties) can be found in an arbitrarily large database of people, and the number of comparisons would only be equal to the number of people in the database.

### III. CODING

Another requirement to the system of comparison was the ability to easily store values of compared properties in a database. Each value is usually stored atomically in its own field (column of the database table) in a database of persons [8]. In such a database each column must be processed individually, but the necessary calculations can be performed even within the SQL query. For more complex structures, such as the record of internal mixing of the questions, data fields cannot be prepared in advance to effectively cover atomically all the possible combinations. The questions may contain an unlimited number of such combinations. For this reason, only one field was allocated to store combinations of random values.

<sup>3</sup>  $sgn$  function leaves only the input value of 0, all other (positive) values are converted to 1

<sup>4</sup> *pattern* is a term for the entity which other entities are compared with [32]

To store multiple values in a single item, it is necessary to encode them. It is possible to use e.g. the *blob* data type enabling the record of text or binary data of unlimited length, or a limited *varchar*<sup>5</sup>. In such items, text strings in a given coding can be stored. Due to the limitation of the symbols for printed form, only 6 bits of each character of a string would be used to encode values, i.e. the range of 64 (2<sup>6</sup>) possible characters. With a binary blob it would then be possible to use the whole range of a byte (2<sup>8</sup> = 256 characters) for individual characters.

Characters for encoding 6-bit values can be chosen arbitrarily, but in general they should be selected from between the 32<sup>nd</sup> and 126<sup>th</sup> character of the ASCII table, because these are all viewable in printed form and they are not a subject of the national coding [9]. This eliminates any problems with the encoding of character set, without any further problems when sending it via URL parameter or in a XML file. The string is easily readable by humans and it is also capable of being written in a non-digitized (analog) form.

Selection of characters for the following examples was inspired by the Base64 format [10]. Because the system conversion methods could not be used for this purpose, their order and additional characters were more appropriately adjusted (see Table 3).

Table 3. The code table for conversion between values (0-63) and characters selected for encoding

0	a	8	i	16	q	24	y	32	A	40	I	48	Q	56	Y
1	b	9	j	17	r	25	z	33	B	41	J	49	R	57	Z
2	c	10	k	18	s	26	0	34	C	42	K	50	S	58	6
3	d	11	l	19	t	27	1	35	D	43	L	51	T	59	7
4	e	12	m	20	u	28	2	36	E	44	M	52	U	60	8
5	f	13	n	21	v	29	3	37	F	45	N	53	V	61	9
6	g	14	o	22	w	30	4	38	G	46	O	54	W	62	-
7	h	15	p	23	x	31	5	39	H	47	P	55	X	63	_

Characters + and / used in the Base64 (see [11]) were replaced by others, because these are subject of coding when it is transferred as the URL parameter [12]. The proposed arrangement of characters is partly legible for humans, because lowercase letters and numbers from 0 to 5 form the first half of the table and the uppercase letters, remaining numbers, and other characters are located in the second half. With a good knowledge of the alphabet, a more precise value can be estimated intuitively out of each character.

Another difference from Base64 is the overall approach towards coding. Base64 losslessly encodes arbitrarily large value to the required number of characters (see the example in Table 4).

Table 4. Process of character encoding from plain “yes” to the Base64 “eWVz”

Data (text)	<b>y</b>						<b>e</b>						<b>s</b>									
ASCII code	<b>121</b>						<b>101</b>						<b>115</b>									
Bit pattern	0	1	1	1	0	0	1	0	1	1	0	0	1	0	1	0	1	1	0	0	1	1
6-bit number	<b>30</b>			<b>22</b>			<b>21</b>			<b>51</b>												
Base64	<b>e</b>			<b>W</b>			<b>V</b>			<b>z</b>												

In contrast, the proposed method is based on the premise, that the value of each property will be stored as just a one character, i.e. the encoded value will be transferred to the range of 0-63. Larger values will be stored in a lossy format and smaller ones will not use the full range of the character, but thanks to this approach individual properties can be directly individually compared.

For conversion of property values (a<sub>i</sub>) to the range of 0-63, each value is first normalized to the range of (0, 1) (n<sub>i</sub>, see Equation 7) and then it is multiplied by the maximum value of this range (63). The result rounded to the nearest integer (in this step there is a loss of accuracy) is converted to a character according to Table 3.

$$n_i = \frac{a_i - \min_i}{\text{range}_i - 1} = \frac{a_i - \min_i}{\max_i - \min_i} \quad (7)$$

This procedure of coding can be applied to all numerical values, including nominal, provided the individual categories are first converted to numbers (indexes). This step is necessary for values bigger than 64, while it is not for discrete values of a lower range. With them, the minimum is sufficient to be subtracted from them and the result can be directly used for encoding without using the entire range. This uniform procedure was selected for the following example and for all types of values. Table 5 shows the process of encoding values of properties of two persons from the previous example.

Table 5. Process of encoding values of properties to characters

Property	Person A				Person B			
	Value	n <sub>i</sub>	0-63	Char	Value	n <sub>i</sub>	0-63	Char
Height	175	0.750	47	P	165	0.694	44	M
Weight	85	0.329	21	v	60	0.231	15	p
Age	35	0.276	17	r	25	0.197	12	m
Sex	2	1.000	63	_	1	0.000	0	a
Color of eyes	5	0.800	50	S	1	0.000	0	a

As shown in Table V, the characteristics of person A are encoded in a “characteristic” text string of “Pvr\_s”, and person B’s to the string of “Mpmaa”. From these strings it is apparent at first glance that these are two different people that do not match in any parameter, including the last two nominal values.

<sup>5</sup> *varchar* – database type for a string of a variable length limited from above [33]

#### IV. COMPARISON OF CHARACTERISTIC STRINGS

Comparing of two characteristic text strings can be done in several ways, or in various stages of coding back to the original value. Various weights must be used to achieve the same result in each of these stages.

The easiest way is the absolute differences of values converted directly from the individual characters (see col. "Code values – difference" in Table 6).

Table 6. The process of decoding individual characters of a text string

Property	Characters		Code values (0–63)		
	A	B	A	B	difference
Height	P	M	47	44	3
Weight	v	p	21	15	6
Age	r	m	17	12	5
Sex	_	a	63	0	1
Color of eyes	S	a	50	0	1
<b>Total</b>					<b>16</b>

The second option are the absolute differences of values normalized to the range of  $\langle 0, 1 \rangle$   $n_i$ , i.e. the previous value divided by 63 (see col. "Normalized values – difference" in Table 7).

Table 7. Back-calculation of the original values ( $a_i$ ) of individual properties (continued from Table 6)

Property	Normalized values (0, 1)			Original value ( $a_i$ )		
	A	B	difference	A	B	difference
Height	0.746	0.698	0.048	174.3	165.7	8.6
Weight	0.333	0.238	0.095	86.0	61.7	24.3
Age	0.270	0.190	0.079	34.3	24.2	10.1
Sex	1.000	0.000	1.000	2.0	1.0	1.0
Color of eyes	0.794	0.000	1.000	5.0	1.0	1.0
<b>Total</b>			<b>2.222</b>			<b>44.9</b>

The third option is to compare the original decoded values ( $a_i$ ) by the means of inverse operation of calculating  $n_i$  (Equation 7) as shown in Equation 8.

$$a_i = n_i \cdot (\text{range}_i - 1) + \text{min}_i = n_i \cdot (\text{max}_i - \text{min}_i) + \text{min}_i \quad (8)$$

Original weights (see Table 1) can be used only in the third stage, i.e. when comparing the original values. However, a one-off conversion of weights saves a need of repeated conversion of coded values to the original ones for each property of each compared subject.

#### V. QUESTION MIX

Initial setting of encoding conditions (metadata) for the specific cases would be performed by someone who would

assess ranges and weights best suited for the particular purpose, or would make experimental measurements and subsequently edit the ranges accordingly. For test questions [13] containing various random elements in various positions of the question, the settings would be very difficult for the creator of questions, in some cases (e.g. for mixing of content position) even impossible. The whole system should thus work completely automatically in the background.

Since the sequence of the same random elements in a question may differ on each occurrence, and some portions may even be completely omitted, fixed identifiers (*id*) were used to detect the elements. On the first processing of a question the metadata (or the data necessary for coding of values) are listed under their identifiers, i.e. types of values and their minimum and maximum<sup>6</sup>. Numeric identifiers<sup>7</sup> of positions (*index*) and values (*val*) are also assigned to these elements (or their items) if necessary. Through them they will then be encoded into the characteristic string. The order of items (individual characters) in the encoded string then determines the order of their record in this list. List of metadata (e.g. see Code 1) is stored in a database directly within the record of the question and, can be expanded if necessary (when adding new elements or uncovering previously inaccessible elements due to random selection).

```
<metadata>
  <versions index="0">
    <ver id="x" val="0" />
    <ver id="y" val="1" />
  </versions>
  <rnd index="1" id="a" type="car" min="10" max="99" />
  <rnd index="2" id="b" type="car" min="1" max="9" />
  <rnd index="5" id="var" type="nom" /> <!-- char A-Z -->
  <mix id="m1">
    <itm index="3" id="i1" />
    <itm index="4" id="i3" />
    <itm index="6" id="i2" />
  </mix>
</metadata>
```

Code 1. Sample of possible record of question metadata to XML

The string containing random values for each use of each question is stored in the database during the generation of the test, along with other statistics for each instance of the question, in a form of a plain text string (*vchar*). An example of the coding of a question values with random elements described by metadata in Code 1 into the string of "bMyb#ha" is shown in Table 8.

<sup>6</sup> this information is directly specified for random values  $\langle \text{rnd} \rangle$  of the range and in other cases they are not required

<sup>7</sup> identifiers specified by creator of questions may not be numeric

Table 8. Sample of encoding of question random values into the string "bMyb#ha"

Element	Value	Coding process	Character	Index
ver	y	$\text{val}["y"] = 1$	b	0
a	72	$\frac{72-10}{99-10} \doteq 0.7 \rightarrow 0.7 \cdot 63 \doteq 44$	M	1
b	4	$\frac{4-1}{9-1} = 0.375 \rightarrow 0.375 \cdot 63 \doteq 24$	y	2
var	H	$\text{index H in } \{A-Z\} = 7$	h	5
m1	i2,i1	$i1 = 1, i2 = 0, i3 = \text{null}$	ba#	3,6,4

If any of the values in the list of a current question mix was omitted due to the random selection, it would still have to be included in the code string, so that its individual positions would always correspond to the same elements. For these cases, a special character # (hash, or sharp) will be used as the equivalent of the *null*<sup>8</sup> value, in the following example. It indicates that the item's value is not represented in the string<sup>9</sup>. In this case there are three basic approaches for comparing this value with others.

- Mark both strings as incomparable.
- Set a zero difference for the given property in the comparison of these specific characters.
- Evaluate the fact that the property exists in one case and not in the other, as the maximal difference. [14]

The choice of the optimal approach depends on the particular application. The third option (maximum difference) was selected for comparing question mix.

#### A. Multiple comparisons

The process of comparing two entities, or their characteristic strings, was explained in the previous chapter. This method can be applied to an arbitrarily extensive list and thus a subject can be found that is most similar or dissimilar to the pattern object. The aim of comparing a question-mix is to determine the difference not only between two subjects but between one entity (generated question) and the group of entities (previously used questions).

If the same tested user has been previously asked the same question more times, then only the last five cases<sup>10</sup> are used for comparing, as it helps to save computational cost. The pattern string is compared with each of them and the total difference is the sum of these values. Individual comparisons, however, do not have the same weight. A similar principle as was the forgetting component at the selection of test questions (see [4]) is used in this case. In this case the weight is only repeatedly lowered<sup>11</sup> by one fifth<sup>12</sup> in order to accelerate the calculations.

#### B. Most different random values

There are more ways to create the most diverse question to its earlier versions. For example, the whole question can be generated several times and the version which is most different from all the previous versions will be used [15]. This will be evaluated by the comparing of their characteristic strings. Respecting individual values during question creation can also be used, which is possible due to their division into individual characters. This method was selected, because of the complexity of repeated parsing of QML. Therefore, since there is no comparison of a whole question but only at the level of individual elements, weights and mutual ranges of these parameters do not have to be taken into account in the calculation.

#### C. Versions

Random variant of a question based on the versions `<versions>` defines a finite number of possibilities, one of which is randomly selected. Individual elements of the question are then shown or hidden, accordingly to which version was randomly selected. In terms of the type of values, versions are *nominal*.

For the nominal values such encoding is used which does not convert the value to the range of 0-63 (unlike the previous example of people comparison), but values are indexed by integers from 0 to the count of versions. The impossibility to include all versions into the calculations if their enumeration contains more than 64 variants is undoubtedly the disadvantageous aspect, but this is only a theoretical possibility. The advantage is the possibility to extend the list of other items at any time (up to 64), without invalidating the data from previously completed testing.

Selection of the version is realised by the means of five random choices from all possible versions. If any of them was not used in the previous five representations of the question, this version is declared as the best and the process is done. However, if all five randomly selected "candidates" were used in the previous five cases, the best of them is determined by the highest total sum of weighted nominal differences from the previous selections (see example in Table 9).

<sup>8</sup> *null* is a special value applicable in databases and some object-oriented programming languages across the data types which indicates that the field of the database record or variable is not set to any value [8]

<sup>9</sup> when encoding to the range of 256, this range can be reduced by one (255) and the last value reserved for the *null*

<sup>10</sup> any number of previous cases can be used for comparison, or even all of them

<sup>11</sup> the time component described in the mentioned source can be also used

<sup>12</sup> the latest variant of the question has a weight of 1, 0.8 for the penultimate, 0.6 for the preceding, etc. and the fifth (the oldest in the selection) has a weight of 0.2

Table 9. Sample of selection of the most diverse random version (out of five) in relation to the five previous selections

Question		Prior versions		Random versions					Weighted nominal difference
Date of use	Weight	Character	Version	1	4	5	3	1	
2015-02-08	1.0	b	1	0.0	1.0	1.0	1.0	0.0	
2015-02-01	0.8	f	5	0.8	0.8	0.0	0.8	0.8	
2015-01-26	0.6	e	4	0.6	0.0	0.6	0.6	0.6	
2015-01-05	0.4	d	3	0.4	0.4	0.4	0.0	0.4	
2014-12-15	0.2	f	5	0.2	0.2	0.0	0.2	0.2	
<b>Total</b>				<b>2.0</b>	<b>2.4</b>	<b>2.0</b>	<b>2.6</b>	<b>2.0</b>	

The example in Table 9 shows the procedure of selecting random version out of five possibilities. All proposed versions (2x1, 3, 4 and 5) were used in the previous five cases. First, the previous uses of questions are sorted in the descending order by the date, and according to this order, weights are assigned to them. In the next step, for each of these selections the relevant character is decoded from the question string to the specific numeric identifier of particular used versions. Subsequently, nominal differences between each previous and proposed versions are calculated (0 for identical, 1 for different) and these results are multiplied by weights reflecting topicality of previous choice. Their sum for each “candidates” is the decisive factor for determining the “winning” version. That is the fourth in this example with the ID number of 3 (val="3"), because its overall difference from the previous five versions has a highest value of 2.6.

D. Random values

Generating random values <rnd> for the question is also repeated five times. All random values from these five (r<sub>j</sub>) are compared (the absolute value of the difference) with values of five previous versions (p<sub>i</sub>), and these differences are multiplied by weights (v<sub>i</sub>) reflecting the ‘age’ of questions (see Equation 9).

$$d_{ij} = |r_j - p_i| \cdot v_i \tag{9}$$

The random value (r<sub>j</sub>) with the highest total sum of weighted differences is included into the question (Σ<sub>i=1</sub><sup>5</sup> d<sub>ij</sub>, see example in Table 10).

Table 10. Sample of selection of the most different random integer value (from 1 to 100) with respect to the five previous values

Question		Prior values		Random values					Weighted difference
Date of use	Weight	Character	Value	5	32	17	98	61	
2015-02-08	1.0	s	29.29	24.3	2.7	12.3	68.7	31.7	
2015-02-01	0.8	γ	89.00	67.2	45.6	57.6	7.2	22.4	
2015-01-26	0.6	1	43.43	23.1	6.9	15.9	32.7	10.5	
2015-01-05	0.4	d	5.71	0.3	10.5	4.5	36.9	22.1	
2014-12-15	0.2	p	24.57	3.9	1.5	1.5	14.7	7.3	
<b>Total</b>				<b>118.7</b>	<b>67.2</b>	<b>91.8</b>	<b>160.3</b>	<b>94.1</b>	

The example in Table 10 shows the procedure of a particular case for selection of the random value in the range from 1 to 100, beginning with the decoding characters to the previous values and ending up with determining the best values from the five possibilities. In this example, this is the fourth (j=4) possibility (98), its total difference from the previous five selections is the highest (160.3).

Random selections from the enumerations are treated as nominal, e.g. for values of type *string* (text) or *char* (character). Their selection is similar to the choice of the optimal version. Even in this case it is not necessary to repeat the selection five times, because if the selected value was not in the previous five selections, it can be used directly and the process of selection for this item is over.

E. Random selection

The procedure of mixing the content of the question <mix> is more complicated [16]. Each item from a mix <item> selection has its own identifier under which (with the prefix of an identifier of the mix or as its sub-element as in Code 1) it is registered in question metadata. A value that is stored below is the position of the item in this mix<sup>13</sup>. If any item is omitted (due to the numerical limitations for the selection), a *null* value (#) is inserted on the position of the item in the string. When an assigned position is compared to a *null* value, the difference is the total count of selected items for the current mix element (becoming a maximum mismatch). When two valid values are compared, their relationship is evaluated as the absolute value of their difference.

Table 11. Example of the procedure of determining the most diverse variant of random selections (4 of 6 items) relatively to the previous five

Prior testing			Random selection																																		
Weight	String	Items positions 1 2 3 4 5 6	V1			V2			V3			V4			V5																						
			1	2	3	0	vΣ	3	-	2	0	1	vΣ	0	2	-	1	3	vΣ	1	2	3	0	-	vΣ	3	2	1	0	-	vΣ						
1.0	#c#bda	- 2 - 1 3 0	4	0	0	2	4	0	10.0	4	4	0	1	3	1	13.0	4	0	0	4	2	3	13.0	4	0	4	4	3	4	19.0	4	0	4	4	3	4	19.0
0.8	bcda##	1 2 3   0 - -	0	0	4	3	0	4	8.8	2	4	4	2	4	4	16.0	1	0	4	4	4	4	13.6	0	0	0	4	4	0	6.4	2	0	2	4	4	0	9.6
0.6	d#acb#	3 - 0 2 1 -	2	4	4	1	4	4	11.4	0	0	4	0	1	4	5.4	3	4	4	4	0	4	11.4	2	4	3	4	1	0	8.4	0	4	1	4	1	0	6.0
0.4	#cb#ad	- 2 1 - 0 3	4	0	4	4	4	3	7.6	4	4	4	4	0	2	7.2	4	0	4	0	1	0	3.6	4	0	2	0	0	4	4.0	4	0	0	0	0	4	3.2
0.2	#bcd#a	- 1 2 3 - 0	4	1	4	0	0	0	1.8	4	4	4	1	4	1	3.6	4	1	4	4	4	3	4.0	4	1	1	4	4	3.6	4	1	1	4	4	4	3.6	
<b>Total</b>									<b>39.6</b>							<b>45.2</b>							<b>45.6</b>							<b>41.4</b>				<b>41.4</b>			

<sup>13</sup> also in the case of coding positions of items of random selection is not resized to the range of 0-63 for the possibility of future extensibility of the selection for other items

Even in this case, the five possible variants of the selection with different positions of the individual items are generated, and these are compared with the previous five. The differences of individual items within mixes are summed and multiplied by the weight reflecting the topicality of previous version (same as in the previous cases). Their sum for each “candidate” of random selection is then the decisive factor for determining the “winning” variant, which is again the highest value of this sum (see Table 11).

The example in Table 11 shows the procedure of a particular case of random selection, beginning with the decoding of previous selections from the fragments of the characteristic strings and resulting in determining the best variant of selection out of five. This is V3, its total difference from the previous five selections is the highest with the value of 45.6. According to this variant, out of the 6 items, the 1<sup>st</sup>, 5<sup>th</sup>, 2<sup>nd</sup> and 6<sup>th</sup> item (in this order) should be included in the selection.

For better understanding of the relationship between the values in Table 11, three related values are framed. These are the 4<sup>th</sup> item from V2 selection (2, i.e., the 3<sup>rd</sup> in order will be registered under the index 4), the same item at the penultimate version of the question (0, i.e., there was a 4<sup>th</sup> item in the 1<sup>st</sup> place) as well as the absolute difference in the intersection of the two coordinates ( $|2-0| = 2$ , i.e., the proposed position of the item in comparison to the penultimate question is shifted by 2 positions).

#### F. User interface

The system represents only one checkbox and two numeric editors in the configuration of test (see Fig. 1) in terms of user interface [17].

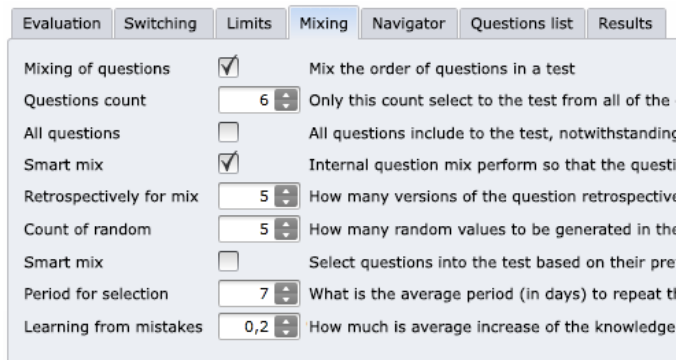


Fig. 1. Example of user interface – Test settings – Mixing tab

The whole process of internal mixing of random elements in questions is enabled or disabled by checking or unchecking of “Smart mix” item. The first numerical value “Retrospectively for mix” ( $n_o$ ) determines how many previous versions of each question for the tested user should be loaded for the comparison process. Weights reflecting the ‘age’ of questions are derived from this value (see Equation 10).

<sup>14</sup> Silverlight is a software plugin for development lavishly furnished internet applications that run within a web browser. It is developed by Microsoft, executed using the plugin which is a smaller version of the .NET framework and written in various languages supported by .NET (e.g. C#). [34]

$$w_{dif.} = \frac{1}{n_o} \quad (10)$$

The second numeric value “Count of random” ( $n_r$ ) determines how many random versions will be generated for each of random elements of the question. The most different from previous versions is selected by comparing with them. This value will be used in a new task for the user. The chance of selecting the really very different value from the previous ones is increased together with this value. However, a higher value means more computational time required to generate questions due to the higher number of necessary comparisons (see Equation 11 – calculating the difference between each of the old version and randomly generated, the sum of these differences for each of the randomly generated versions and comparing of these sums).

$$n_{operations} = 2 \cdot n_o \cdot n_r + n_r \quad (11)$$

#### G. Future application development

User part of the application (testing and administration interface) is created as the Rich Internet Application (RIA) at the Silverlight<sup>14</sup> technology. It allows to an application to use it anytime and anywhere without installing, updating, configuration or own server. A web browser and this plugin is only needed.

However, the modern trend is websites without any plugins (Java, Flash, Silverlight, etc.) and creating of the whole application logic only with HTML 5 and JavaScript. Microsoft had promised the support for Silverlight until 2021, but in a new browser Edge with it no longer count<sup>15</sup>.

However, Silverlight is a technology based on XAML and C# (or any other .NET language), as well as so-called universal applications (UWP) for Windows 10 [18]. Migration of Universal testing environment to this new technology is therefore possible solution. This new version will function on any PC, tablet and mobile with Windows 10, including XboxOne and HoloLens (see [19], p. 17–19). Windows Store<sup>16</sup> would care about its distribution and updates [20]. The web service would be liable to remain as a cloud-based without changes of server part.

Migration of Universal testing environment to UWP should occur before the Microsoft ends the planned support for the Silverlight.

## VI. FURTHER USE

Identifying and comparison of objects properties through the characteristic text strings can be used in many other areas.

#### A. Similar alternatives of the product in eShop

Modern electronic online stores provide a function for comparison of properties of products from the same category. Usually, these systems can also find and offer to the customer

<sup>15</sup> Microsoft Support Lifecycle for Silverlight: <https://support.microsoft.com/en-us/lifecycle?c2=12905>

<sup>16</sup> Microsoft Windows Store: [www.windowsstore.com](http://www.windowsstore.com)

an alternative for the current product, i.e. find the most similar products in the category.

If values of properties of individual products are stored in the database according to the classic rules of normalization [21], then the data model has a structure similar to that of Fig. 2. Each category of products defines a set of properties that can be further expanded by subcategories. Individual products sets a specific value to each property in a coupling table.

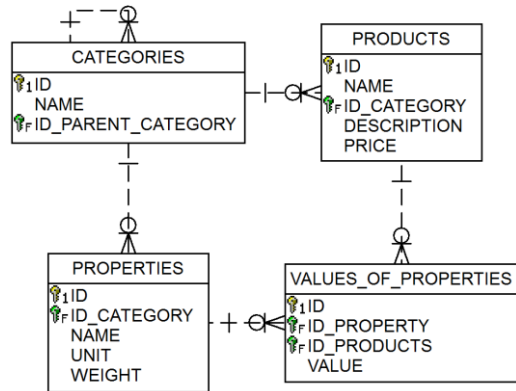


Fig. 2. Illustrative data model for structured records of values of properties of products in classic eShops

All the properties of these compared products must be loaded for creating a table with a list of comparing properties of selected products. The final table will be build up by the application or by using of complicated PSQL<sup>17</sup>. Finding the most similar product then works with all the properties of all products in the same category. The database system receives to each of them by a relatively complex procedure [22].

The structure of the database could significantly simplify by the use of encoding property values to the characteristic text strings (see Fig. 3). All values of all properties of the product would be encoded within a single data field (column) of database table of products as text (varchar) or binary value. Definition of individual properties could be saved in a separate table, as in the previous case (see Fig. 2), or in a single data field in the categories table, like as the definition of questions' metadata (see Code 1).

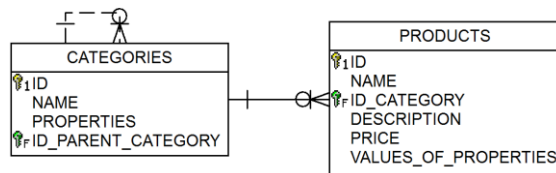


Fig. 3. Illustrative data model for structured records of values of properties of products in eShops through characteristic text strings

Simple selection from a single database table of products with limitations for that category would suffice in this case instead of complicated SQL query (see Code 2).

```
select * from PRODUCTS where ID_CATEGORY = @idCat;
```

Code 2. Sample SQL query to select the data required for find alternative products from the same category

Data from the database can even be loaded directly into the desired object through the Entity Framework [23]. This object is a dictionary in this case. The product ID is the key of this dictionary and the value is product's characteristic text string which contains encoded values of its properties (see Code 3).

```
using (var ctx = new DbContext())
var otherProducts = ctx.PRODUCTS
    .Where(p => p.ID_CATEGORY = idCat)
    .ToDictionary(k => k.ID,
        v => v.VALUES_OF_PROPERTIES);
```

Code 3. Sample code for retrieve data from a database using LINQ query with lambda expressions (C#)

The most similar alternatives to a particular product from loaded data could then be determined by the application layer through a simple algorithm (see Code 4). By transfer maxima computations to the application layer is reduced the database server workload, allowing easier scalability of the computing power of the whole system [24].

```
public IDictionary<int, double> CalcDifferences(
    string pattern, Property[] properties,
    IDictionary<int, string> otherProducts)
{
    var result = new Dictionary<int, double>();
    foreach (var key in otherProducts.Keys)
    {
        string op = otherProducts[key];
        double totalDiff = 0;
        for (int i = 0; i < pattern.Length; i++)
        {
            double diff = Math.Abs(pattern[i] - op[i]);
            if (properties[i].Type == PropType.Nominal)
                diff = Math.Sign(diff);
            totalDiff += diff * properties[i].Weight;
        }
        result[key] = totalDiff;
    }
    return result;
}
```

Code 4. Sample code of method for calculating the differences from other products (C#)

Method in the Code 4 receives as an input the characteristic text string with encoded properties of the pattern product (*pattern*), information (type and weight) about individual properties in the given order (*properties*) and a list (dictionary) of other products properties (*otherProducts*) loaded by the

<sup>17</sup> PSQL – Procedural SQL - is the Firebird programming language used in stored procedures, triggers and executable blocks, see [33], p. 303–315



Code 3. The method calculates the weighted differences for each of the compared products and return it again as a dictionary with ID of the products and values of their differences from the pattern product.

When using the code table would, of course, was necessary to convert the characters to numbers through special functions. In this example, the characters are converted to numeric values according to the ASCII<sup>18</sup> table.

The resulting dictionary can then be sorted e.g. through LINQ by method *OrderBy* and choice can be limited e.g. only to 5 most similar products by method *Take* (see Code 5).

```
var alternatives = CalcDifferences(pattern, props, otherProds)
    .OrderBy(p => p.Value).Take(5);
```

Code 5. Sample code for ascending sort and a choice of five alternatives the most similar of the product on the basis of similarities calculated by method in Code 4

Function to calculate the weighted difference between the two text strings it being possible to directly implement into the database system. E.g. UDF<sup>19</sup> exist for this purpose in RDBS Firebird. The differences between the products could be determined and the data according to this value directly sort already within the SQL query. In the event that the value of properties (all) were stored as a binary value, a computation of the total difference could be even faster, because the calculations can take place at the binary level.

### B. License plate

The license plates of cars are consist of letters and numbers. Most of camera systems can automatically read these license plates by image analysis [25]. Only a region where the owner of the vehicle resides is encoded in the license plate code and the remaining characters are random [26]. The basic condition is that any combination of characters must be unique and the vehicle could be traced by the vehicle registry database.

Some of characters in the license plate could be used to encode other information about the vehicle. For example, it could include the identification of the manufacturer of the vehicle, body color, year, type of vehicle, engine, etc. Multiple values simultaneously could be encoded into one character.

If that were the case, then it would be much easier to uncover a vehicle with stolen license plates, even without internet connection (offline) e.g. with the help of applications in smart mobile phone. This basic checks could then be carried out by persons without access to the central registry of vehicles.

In the case of identification of the vehicle based on the description of a witness who does not know its license plate, part of it could be inferred based of his description of the car.

### C. Genetic algorithms

Genetic algorithms could represent another area of use of the characteristic text strings. *Genetic algorithm is a heuristic scientific method based on Darwin's biological evolutionism*

[27], which has been widely applied to solve high dimensional optimization problem for parameter optimization in engineering and science areas, such as building construction [28] and biotechnology [29]. [30]

Genetic algorithm encodes individual cases to a so-called chromosomes composed of individual genes when finding an optimal solution [31]. These chromosomes are crossed over among themselves, their genes are mutated, or they are reproduced into the next generation.

For their interpretation in computer memory could be used characteristic text strings. They therefore constitute chromosomes and their individual characters would be their genes. Most operations of genetic algorithms would be greatly facilitated thanks to the above-described method of encoding and comparing of elements.

## VII. CONCLUSION

The characteristic text strings allow us to encode values of properties of different objects into legible characters and to mutually compare their similarities. The whole process was demonstrated on comparing the basic characteristics of two different people. The procedures and ways of practical use of such strings were also presented used to generate more diverse, rather than purely random test questions for the same tested user. The same procedure can also be used for various tested users during mass preparation of questions at the same time, for a single IP address or a computer lab, as a precaution against possible cheating. Other possible uses were outlined, including various ways of implementation of the entire process or its individual parts.

Thanks to the code strings, database structures can be simplified. Support for searching and comparing these strings directly by SQL functions should not be difficult to implement in some database systems.

Comparing, however, is not always performed on the large database of different subjects, but thanks to the ease of portability of characteristic strings can be realised individually "in the field", e.g. using a mobile phone. In addition to text characters for storing values, it is also possible to use other data structures, e.g. barcode or QR code.

Thanks to the separation of the individual properties to the individual characters, it is also possible to compare only certain parts of them, independently of the rest without decoding the whole string.

Areas of application are broad, and this principle can be used wherever it is necessary to store and compare the cardinal or nominal characteristics. These may include e.g. parameters of goods, searching for people, vehicles, dating etc. Strings encoding characteristics of objects with possibility of fast mutual comparisons are also a very important part of genetic algorithms, where this approach could also be applied.

<sup>18</sup> ASCII – American Standard Code for Information Interchange

<sup>19</sup> UDF – User-Defined Functions – functions that are not internal to the engine, but defined in separate modules, see [33], p. 328–334

## REFERENCES

- [1] P. Voborník, "Universal Testing Environment as an External Tool of Moodle," in *10th International Scientific Conference on Distance Learning in Applied Informatics (DiVAI 2014)*, Štúrovo, Slovakia: Wolters Kluwer, pp. 215–225, 2014, ISBN 978-80-7478-497-2. Available: <http://download.petr.vobornik.cz/docs/clanky/2014-divai.pdf>
- [2] M. MacDonald, *Pro Silverlight 5 in C#, 4th ed.*, New York: Apress, 2012, ISBN 978-1-4302-3479-1.
- [3] P. Voborník, "Universal Testing Environment," Ph.D. dissertation, Faculty of Informatics and Management, University of Hradec Králové, Hradec Králové, Czech Republic, 2012. Available: <http://download.petr.vobornik.cz/docs/disertace.pdf>
- [4] P. Voborník, "The computing model for intelligent selection of the testing questions," in *The proceedings of the 14th international conference MEKON 2012*. Ostrava: VŠB-TUO, 2012, ISBN 978-80-248-2552-6. Available: <http://download.petr.vobornik.cz/docs/clanky/2012-mekon.pdf>
- [5] M. Meloun and J. Militký, "Přednosti analýzy shluků ve vícerozměrné statistické analýze," in *Sborník přednášek z konference Zajištění kvality analytických výsledků*, Medlov: 2 THETA, pp. 29–46, 2004, ISBN 80-86380-22-X.
- [6] Š. Hubálovský and E. Milková, "Modeling of a real situation as a method of the algorithmic thinking development," in *Advanced Educational Technologies, Proceedings of 6th WSEAS/IASME International Conference on Educational Technologies (EDUTE'10)*, WSEAS Press, Kantoui, Sousse, Tunisia, May 3–6, pp. 68–72, 2010, ISBN 978-960-474-186-1, ISSN 1790-5109.
- [7] F. Pavelka and P. Klímeck, *Aplikovaná statistika*, Zlín: FaME, 2000, ISBN 80-214-1545-2.
- [8] P. O'Neil, *Database: Principles Programming Performance*, San Francisco: Morgan Kaufmann, 2014, ISBN 9781483184043.
- [9] Š. Hubálovský and P. Hanzalova, "Modeling, simulation and visualization of automatic cryptanalysis of the short monoalphabetic substituted cipher text," *International Journal of Mathematics and Computers in Simulation*, vol. 7, is. 2, pp. 134–143, 2013, ISSN 1998-0159.
- [10] L. L. Wen, X. Z. Ruo, K. Jian, T. Ling and H. C. Guang, "A Design of Improved Base64 Encoding Algorithm Based on FPGA," *Applied Mechanics and Materials*, vol. 513–517, pp. 2220–2223, Feb. 2014, ISSN 1662-7482.
- [11] D. Ashley. (2010, Sep. 20). Obfuscation used by an HTTP Bot. Information Security Office [Online]. Available: <http://security.utexas.edu/consensus/Obfuscation.pdf>.
- [12] Š. Hubálovský, "Remote Contact Learning of Programming in Distance Study," in *10th International Conference Efficiency and responsibility in education 2013: proceedings (ERIE 2013)*, Praha, Česká zemědělská univerzita, pp. 210–217, 2013, ISBN 978-80-213-2378-0.
- [13] E. Milková, "Multimedia applications as an effective complement to the educational process," in *Advanced Educational Technologies, Proceedings of 6th WSEAS/IASME International Conference on Educational Technologies (EDUTE'10)*, WSEAS Press, Kantoui, Sousse, Tunisia, pp. 62–67, May 2010, ISBN 978-960-474-186-1, ISSN 1790-5109.
- [14] E. Milková and O. Kořínek, "Students' programming capabilities evaluation," in *Efficiency and Responsibility in Education 2013*, Czech University of Life Sciences Prague, pp. 434–440, 2013.
- [15] Š. Hubálovský, "Research of Methods of a Multidisciplinary Approach in the Teaching of Algorithm Development and Programming," in *9th International scientific conference on distance learning in applied informatics (DiVAI 2012)*, Nitra, Univerzita Konštantína Filozofa, pp. 147–156, 2012, ISBN 978-80-558-0092-9.
- [16] Š. Hubálovský and J. Šedivý, "Algorithm Development and Computer Simulation of Position Order Decoding of Mastermind Board Game," *Applied Mechanics and Materials*, vol. 333–335, pp. 1353–1356, Jul. 2013, ISSN 1662-7482.
- [17] R. Němec, M. Hubálovská and Š. Hubálovský, "User Interface of System SMPSL," in *Communications and information technology (CIT 2014)*, Salem: North Atlantic University Union, pp. 324–329, 2014, ISBN 978-960-474-361-2.
- [18] M. Halsey, *Windows 10 Primer: What to Expect from Microsoft's New Operating System*, Apress, Apr. 2015, ISBN: 978-1-4842-1047-5.
- [19] A. Badju and D. Lundberg, *Shopping Using Gesture Driven Interaction*, Master's Thesis, Department of Design Sciences, Lund University, Sweden, May 2015.
- [20] V. Diba and C. Wagner, "Success within App Distribution Platforms: The Contribution of App Diversity and App Cohesivity", in *System Sciences (HICSS), 2015 48th Hawaii International Conference*, IEEE, pp. 4304–4313, Jan. 2015, ISSN 1530-1605, DOI 10.1109/HICSS.2015.515.
- [21] W. Kent, "A simple guide to five normal forms in relational database theory," *Communications of the ACM*, vol. 26, is. 2, New York, pp. 120–125, Feb. 1983, DOI 10.1145/358024.358054.
- [22] M. Hubálovská and Š. Hubálovský, "Implementation of the Systems Approach in Mathematical Modeling, Dynamic Simulation and Visualization Using MS Excel Spreadsheet," *International Journal of Mathematics and Computers in Simulation*, vol. 7, no. 2, pp. 267–276, 2013, ISSN 1998-0159.
- [23] A. Adya, J. A. Blakeley, S. Melnik and S. Muralidhar, "Anatomy of the ADO.NET entity framework", in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, ACM, Beijing, China, pp. 877–888, Jun. 2007, ISBN: 978-1-59593-686-8, DOI 10.1145/1247480.1247580.
- [24] I.-H. Chung and J. K. Hollingsworth, "Automated cluster-based Web service performance tuning", in *Proceedings of the 13th IEEE International Symposium High performance Distributed Computing 2004*, IEEE, pp. 34–44, Jun. 2004, ISBN 0-7695-2175-4, ISSN 1082-8907, DOI 10.1109/HPDC.2004.1323484.
- [25] S.-L. Chang, L.-S. Chen, Y.-C. Chung and S.-W. Chen, "Automatic license plate recognition", *Intelligent Transportation Systems, IEEE Transactions*, IEEE, pp. 42–53, Mar. 2004, ISSN 1524-9050, DOI 10.1109/TITS.2004.825086.
- [26] Act No. 56/2001 of Coll., "On the Conditions for the Operation of Vehicles on the Road Network", *Collection of Laws*, Czech Republic, 2001.
- [27] J. A. Foster, "Evolutionary computation", *Nature Reviews Genetics*, vol. 2, pp. 428–436, Jun. 2001, ISSN 1471-0056, DOI 10.1038/35076523.
- [28] M. Manzan, "Genetic optimization of external fixed shading devices", *Energy and Buildings*, vol. 72, pp. 431–440, Apr. 2014, ISSN 0378-7788, DOI 10.1016/j.enbuild.2014.01.007.
- [29] K. H. Kucharczyk, R. L. Crawford, A. J. Paszczyński, T. Soule and T. F. Hessa, "Maximizing microbial degradation of perchlorate using a genetic algorithm: media optimization", *Journal of Biotechnology*, vol. 157, issue 1, pp. 189–197, Jan. 2012, ISSN 0168-1656, DOI 10.1016/j.jbiotec.2011.10.011.
- [30] L. Jiang, H.-H. Xiao, J.-J. He, Q. Sun, L. Gong and J.-H. Sun, "Application of genetic algorithm to pyrolysis of typical polymers", *Fuel Processing Technology*, vol. 138, pp. 48–55, Oct. 2015, ISSN 0378-3820, DOI 10.1016/j.fuproc.2015.05.001.
- [31] J. Šedivý and G. Garant, "Computer Modeling, Simulations and Design in the Engineering – Research of Selected Indicators," *International journal of education and information technologies*, vol. 7, is. 3, North Atlantic University Union, pp. 98–106, 2013, ISSN 2074-1316.
- [32] V. Strnadová, *Interpersonální komunikace*, Hradec Králové: Gaudeamus, 2011, ISBN 978-80-7435-157-0.
- [33] P. Císař, *InterBase / Firebird - Tvorba, administrace a programování databází*, Brno: Computer Press, 2003, ISBN 80-7226-956-1.
- [34] T. Lammarsch, W. Aigner, A. Bertone, Miksch, S., Turic, T. and Gärtner, J., "A Comparison of Programming Platforms for Interactive Visualization in Web Browser Based Applications," in *International Conference Information Visualisation*, Washington DC: IEEE Computer Society, 2008, ISBN 978-0-7695-3268-4.



**Petr Voborník** was born in the Czech Republic in 1982. He received a master degree (Ing.) in Information Management and a doctoral degree (Ph.D.) in Information and Knowledge Management from the University of Hradec Králové in 2006 and 2012. He is now an Assistant Professor at the Department of Informatics, Faculty of Science, University of Hradec Králové. His current research interests include developing new algorithms and mini-languages for optimization of electronic testing for his Universal Testing Environment. He also participates in several researches as a programmer, he teaches and popularizes programming and he creates independent applications and games.