Expanding on the Concept of Quality

Catalin Silviu Nutu [1)], Tiberiu Axinte [2)]

[1)] *Constanta Maritime University, Constanta, Romania*
[2)] *Research and Innovation Center for Navy, Constanta, Romania*

"The control of shape is one thing,
but the way to victory cannot be only a single one"
Sun Bin

CONTENT

AN ENCIPHERING MODEL ACCORDING TO SHANNON'S THEORY

Based on the weighted addition secrecy method presented, corresponding to formula (1) above, one can generate a certain specific encryption method, in the way it is presented in the following model.
Let us set the number of enciphering operations to k = 1100. Let us set the number of encoding transformations to 10, encoding transformations which can be randomly ordered: $T_{1,}T_{2,\dots,}T_n$ with n=10. For both, clarity and simplicity reasons, let us consider that each transformation obeys the rule: $T_i = \alpha_i I_i + \beta_i$, with α_i and β_i integers and I_i the inputs of the enciphering transformations. Obviously, one can choose randomly generated pairs (α_i, β_i).
Let us also set the probability related to the occurrence of each of the ten above transformations:

$$p_1 = \frac{1}{55}, p_2 = \frac{2}{55}, \dots, p_i = \frac{i}{55}, \text{ for } i = \overline{1,n}. \quad (6)$$

The number of occurrences for each transformation is then given by the formula: $T_i = p_i k$. One can then randomly order the n above transformations by randomly assigning the indices for each of the ten transformations.
Then the following triangle matrix can be generated for the transformations randomly chosen as above, so that their probability obeys the rule in formula (6):

$$\begin{matrix} T_1 & T_2 & \dots & \dots & \dots & \dots & \dots & T_n \\ & T_2 & \dots & \dots & \dots & \dots & \dots & T_n \\ & & T_3 & \dots & \dots & \dots & \dots & T_n \\ & & & & & \dots & \dots & T_n \end{matrix} \quad (7)$$

In this way, by going through each line of the above matrix 20 times, it is ensured that the probability for each transformation is according to formula (6). According to a key which is a natural number chosen between 1 and 1100 and the previously generated matrix (7), the corresponding transformation it is chosen by going through the elements of this matrix, until the corresponding transformation is reached.
One can either find the index i of the transformation to be performed on the output by using the interval to which the key number belongs to, or the position in the matrix can be determined and then the exact position in the corresponding line is also found. For example, in this model presented, for a key number between 1 and 200, the corresponding transformation has the index i, with $i = \left[\frac{KEY}{20}\right] + 1$, where [x] I stands for the integer part of the number x. Further on, if KEY belongs to interval (201, 380], then the index i of the respective transformation to be performed on the input is $i = \left[\frac{KEY}{20}\right] + 8$, a.s.o.
The transmitted information can be decoded at the other end of the transmission channel, using the reverse transformation, namely:

$$I_i = \frac{T_i - \beta_i}{\alpha_i} \quad (8)$$

One can use either the same key number to encode an entire certain message or a vector of key numbers can be used to encode each symbol transmitted, using a different key, as it will be presented further on. Next section comprises in a few examples, by means of C++ algorithms the theory presented so far. These C++ subroutines address and implement different approaches related to the theory example presented in section 2 of this paper.

C++ ALGORITHMS FOR THE IMPLEMENTATION OF THE PRESENTED ENCIPHERING MODEL

The next subroutine actually solves the problem of choosing the key number in certain intervals, by calculating the exact position of the transformation using the mathematical concept of integer part of a real number x, [x], as already presented in the section above. It also uses the extended model with a vector of key numbers instead of using just only one single key used for encryption.

```
#include <iostream>// encryption for NR in [0, 1099]
using namespace std;

int i[26];
int j;
int pos[26];
int p, q;
int alpha[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int beta[10] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
int NR[26] = {21, 22, 25, 27, 47, 223, 227, 235, 247, 255, 256,
257, 381, 389, 391, 399, 417, 418, 419, 541, 543, 545, 547, 549, 550, 1077 };
//Set of encryption keys
int I[26] = {1, 2, 3, 4, 5, 6, 7, 8, 9 , 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26};// Input to be encrypted
int T[26];
int O[26];// Encrypted output
int main(void) {
for (j=0; j<26; j++)
{
   pos[j] = int(NR[j]/20)+1;

   if (pos[j]<=10) {i[j] = pos[j];}
   else if ((10<pos[j]) && (pos[j]<=19)) {i[j] = (pos[j] - 10) + 1;}//{i =
((int(NR/20)+1)-10)+1;}
   else if ((19<pos[j]) && (pos[j]<=27)) {i[j] = (pos[j] - 19) + 2;}
   else if ((27<pos[j]) && (pos[j]<=34)) {i[j] = (pos[j] - 27) + 3;}
   else if ((34<pos[j]) && (pos[j]<=40)) {i[j] = (pos[j] - 34) + 4;}
   else if ((40<pos[j]) && (pos[j]<=45)) {i[j] = (pos[j] - 40) + 5;}
   else if ((45<pos[j]) && (pos[j]<=49)) {i[j] = (pos[j] - 45) + 6;}
   else if ((49<pos[j]) && (pos[j]<=52)) {i[j] = (pos[j] - 49) + 7;}
   else if ((52<pos[j]) && (pos[j]<=54)) {i[j] = (pos[j] - 52) + 8;}
   else if ((54<pos[j]) && (pos[j]<=55)) {i[j] = (pos[j] - 54) + 9;}
}

for (j=0; j<26; j++)
{
   cout << "nth i is :" << " "<< i[j] << endl;
}

for (j=0; j<26; j++)
{
   O[j] = alpha[i[j-1]]*I[j] + beta[i[j-1]];
   cout << "The nth output O is :" << O[j] << endl;
}
return 0;
}
```

CONCLUSIONS

This paper shows that in order to estimate future quality score for a certain product, one does not necessarily need to measure the score of each quality feature at each step. It is only necessary, based on previous evolutions, to approximately determine the coefficients of a given set of possible quality transformations most likely to occur in the future. This can also be viewed as a quality prediction system which could be used for all sorts of complex products, having multiple quality features.

Out of simplicity and clarity reasons, the models presented in this paper are based on and use a determined set of T transformations. This presented model, however, can be extended using high performance computer technology, for very large sets of transformations, each next transformation deriving out of the previous one, according to certain laws and functions.