The application of the theory of dynamic systems to software quality estimation

A V Kopyltsov

Department of Computer Science, Saint Petersburg State Electrotechnical University,
5 Professor Popov Street, St. Petersburg, 197376, Russia
Department of Higher Mathematics and Mechanics, Saint Petersburg State University of Aerospace Instrumentation, 67 Bolshaya Morskaya Street, St. Petersburg, 190000, Russia

E-mail: Kopyl2001@mail.ru

Abstract. Hardware and software are considered from the point of view of the theory of dynamic systems. It has been shown that a program may be considered as dynamic system placed into the phase space of the program variables. The theory of dynamic systems as applied to programming allows to classify programs revealing the important quality indices of the program products which are related to the basic concepts of the ergodic theory as well as to that of information and informatics. As a program is run, possibilities (complexity) of the computer may be evaluated and important quality indices of the hardware may be appreciated from the point of view of the programming itself. This approach lets also separately evaluate the quality of both hardware and software.

1. Introduction

In the last few years, computers have acquired a great importance for the control processes (economics, manufacture, transport, etc.), information gathering and storing as well as for scientific researches and techniques. It is not easy in fact to find a field of human activity in which computers are not used. From year to year, the field of application of the computers is growing and the demands placed upon its quality are becoming more exacting.

Computation quality is made of that of hardware and of that of software. The problem of providing the quality is nowadays characterized by a poor methodological base, absence of high-efficient instrumentation for adjusting, testing, assessment and control of the quality of the programs as well as by an insufficient information support. Actual approaches of assessment of hardware and software quality (mathematical statistic, qualimetric, non-acute, etc.) permit to appreciate some quality indices related to the hard- or software [1-10]. Nevertheless, there is no unified approach allowing to obtain at the same time and from the point of view of the programming itself the assessment of the quality of the hard- and software or both to classify programs and to reveal the relation existing between the quality indices and the fundamentals of the modern physics. Such an approach allowing to fulfill the above tasks seems to be the one based on the theory of dynamic systems [11].
2. Results and discussion

Fortran, Cobol, PL/1, Algol, Pascal, C, Ada and many other languages which are not so spread refer to the algorithmic or von - Neumannan ones. The principle of algorithmic programming may be reduced to the two following basic ideas, i.e. to that of appropriation operator and of control transfer [12]. Let us assume that a program is written by means of one of the above algorithmic languages to compute within the sub-set $M$ of the field of real numbers. We consider the number of the elements of the $M$-set as equal to $MM$. Let in the program be one real variable. Then the appropriation operator preset for the set $M$ will behave as follows:

$$A : X_1 \rightarrow X_2,$$

i.e. it will transfer the real number $X_1$, to that of $X_2$ pertaining both $X_1$ and $X_2$ to $M$. The set of all the appropriation operators $\{A_i\}$, preset for the $M$-set of $MM$-capacity provides for partition to equivalent classes. We consider that the two operators

$$Ax : X_1 \rightarrow X_2$$

and

$$Ay : Y_1 \rightarrow Y_2$$

pertain to the same class if $N_{X_1}-N_{X_2}$ is equal to $N_{Y_1}-N_{Y_2}$ as related to the module $MM$, i.e.

$$N_{X_1}-N_{X_2} \equiv N_{Y_1}-N_{Y_2} \pmod{MM}.$$ 

Let $NM$ be the maximum value applied to computer. We consider a space having the measure $(\sigma, \mu)$, where $M$ is a space using as its elements the real numbers applied to computer, i.e. $\sigma$ is the $\sigma$-algebra of the whole sub-set $M$, $M=\{-NM,...,0,...,NM\}$ and $\mu$ is measure of normalization of $M$, i.e. $\mu(M)=1$ and all the points are equiprobable.

Let $T$ be a cyclic substitution for $M$. Then, according to [13], is an ergodic transformation $T$ generates the monoparametric group of automorphisms $\{T_i\}$ of the space having the measure $(\sigma, \mu)$, i.e. it generates the dynamic system $\{T_i\}$ for $(M, \sigma, \mu)$ [14]. Thus, the following affirmation is valid.

Affirmation 1. The set of all the appropriation operators assigned for the set $M$ permit partition to equivalent classes for which a group structure (monoparametric) may be introduced. This dynamic system (group of automorphisms) is ergodic.

If the program includes more than one variable, the direct product of the dynamic systems must be taken thus enunciating again the Affirmation 1 as follows.

Affirmation 2. In the case $N>1$ of variable sets of all the appropriation operators assigned for the set $M \times ... \times M$ ($N$ co-factors), partition to equivalent classes is admissible to assign a dynamic system being equal to the direct product of the dynamic systems acting on the space $(M, \sigma, \mu)$. This dynamic system is ergodic.

Since the program may be considered as a sequence of appropriation and control operators [12], from the Affirmation 2, the following conclusions may be obtained:

- to a certain run of an arbitrary program a certain space automorphism $(M, \sigma, \mu)$ corresponds,

- to every space automorphism $(M, \sigma, \mu)$ not more than one even number of the programs run corresponds.

Thus, programs may be considered as dynamic systems and, consequently, to the software analysis methods and results obtained from the theory of dynamic systems may be applied. In particular, application of the theory of dynamic systems in programming allows to classify the programs to reveal the important indices of the program product quality which are related to the fundamentals of the modern physics. Besides, according to the program run, conclusion may be done on the computer complexity being so an important index of hardware quality from the point of view of the programming itself, and a separate assessment of the hard- and software quality may be carried out. Let us consider each one of the above statements in detail.

Since program may be considered as a dynamic system, it is natural to apply such well-known methods of spectral analysis as the rapid Fourie transformation and the autocorrelation function. To
the program run, trajectory built in the \( N \)-dimensional phase space of variables (where \( N \) is the program variables number) is juxtaposed. Methods of spectral analysis allow to reveal the periodic, quasi-periodic or chaotic modes of the trajectory behavior in the \( N \)-dimensional space. In the last case, concept of the fractal dimension is important. If the trajectory is rather complicated, its fractal dimension may be assessed by different ways [15].

In paper [16], it has been shown that \( d_G \leq d_I \leq d_C \) being the values of \( d_p \) (point-to-point dimension) , \( d_C \) (correlation dimension) , \( d_I \) (informational dimension) and \( d_C \) (capacity dimension) close together as a rule.

As quality indices of the program product, fractal dimensions and information entropy may be used. The fractal dimension can be considered as the order of the program complexity (as the dimension is greater, the program is more complicated). The fractal dimension lets introduce a relation of the order \( "<" \) for the set of all the programs. If \( A \) and \( B \) are programs, then \( A < B \) if the fractal dimension \( A \) is not greater than the fractal dimension \( B \). It allows to introduce classification of the programs. For instance, class 1 is the fractal dimension from 0 to 1 being the class that from 1 to 2,

amounting the class 3 to the fractal dimension from 2 to 3, etc.

Information entropy may be considered, same as the fractal dimension, as the measure of complexity of the program (in bits). Definition of such characteristics as dimension and entropy may be done either in limit set (attractors) or the above sets may not be considered as all in the phase space being the whole diagnostics built on the mere processing of a specific (rather long.) time realization of the program under study [16].

If two programs are written on the base of the same algorithm, they may be compared one to another after definition of their dimensions and entropy, i.e. it may be found which one of the algorithm realization is the most complicate (having the greatest dimension). Thus, dimension and entropy may be considered as quality indices.

The information entropy \( IE \) may be considered as a measure of the random quantity in an experiment made of one single run of the program and called entropy of this experiment. \( IE \) measures also the indeterminacy quantity into this experiment, i.e. the indeterminacy quantity before the program run a related to what its result will be [13]. Since when determining the fractal dimensions \( d \), a limit should be taken for \( N \) and \( \varepsilon \) approaching infinity and zero, respectively, being usually \( N_0 \) in the programs a finite value, \( d \) must be used cautiously (only when \( N \) is of the order \( 10^3 \) - \( 10^4 \) and more).

As the respective estimations have shown it, the information entropy \( IE \) is more stable when the input parameters vary a compared with the fractal dimensions. Therefore, when \( N_0 \) has low values, it is more reliable to use \( IE \) than \( d \). If the range of variation of the program input parameters is known, a random sample of this range parameters may be done averaging \( IE \) according to the range of variation of the parameters. The averaged \( IE \) value may be obtained by so doing.

Calculations have been done for a program solving the system \( n \) of algebraic equations according to the Gauss method (for \( n=2 \) and \( 3 \)) [12]. Thus, for the system 2 of equations having

\[
\begin{align*}
a_1x + b_1y &= c_1, \\
a_2x + b_2y &= c_2,
\end{align*}
\]

we have 22 real variables and 74 points within a 22-dimensional phase space of the program variables.

Since the number of the points included into the phase space is not great (being equal to 74 what is less than \( 10^3 \)), it is not desirable to use the fractal dimension as a quality index (although the program permits to evaluate it, it is not faithful). Therefore, the information entropy \( IE \) has been used which amounts to about 5 bits when \( a_1=2, b_1=1, b_2=-2, c_1=9, c_2=-2 \). When the input parameters have other values, figures close to them are obtained amounting \( IE \) from 4 to 5 bits.

It is analogous for the system 3 of equations

\[
\begin{align*}
a_1x + b_1y + c_1z &= d_1, \\
a_2x + b_2y + c_2z &= d_2, \\
a_3x + b_3y + c_3z &= d_3,
\end{align*}
\]
where the number of the program variables accounts for 39 being the number of the points contained within the 39-dimensional phase space equal to 192 what it also less than $10^3$. Therefore, evaluation of $IE$ has been done according to the same reasons amounting to 7 bits when $a_1=7$, $a_2=6$, $a_3=8$, $b_1=6$, $b_2=3$, $b_3=8$, $c_1=8$, $c_2=7$, $d_1=14$, $d_2=3$, $d_3=12$. Taking other values of the coefficients, similar figures are obtained being $IE$ equal from 6 to 7 bits.

Thus, $IE$ of the system 3 of equations is greater than that of the system 2 of equations. Therefore, the first program is more complicated than the second one.

According to the program run, the conclusion may be done as related to the computer complexity as well as an important index of the hardware quality may be found from the point of view of the programming itself. Let $R$ actually be set of all the real numbers being $A$ an algorithm realized by means of the two computers $C_1$ and $C_2$ in the form of the two similar programs $PR_1$ and $PR_2$, where $P$ is the number of the variables of the programs, and $R^P = R \times ... \times R$ ($P$ - co-factors). We consider $PR_1$ and $PR_2$ as the following maps

$$ PR_1: R^P \rightarrow R^P, \quad PR_2: R^P \rightarrow R^P. $$

Let $M$ be a set of real numbers equal to the union of the sets of real number which are used by all the actual computers. Let the number of the element contained within the $M$-set be equal to $N$. We consider then a space having it measure equal to $(M, \sigma, \mu)$, where $\sigma$ is $\sigma$-algebra of all the $M$-subsets being, $\mu$ a measure normalized to $\sigma$, i.e. $\mu(M) = 1$ and all the points of the $M$-space are equiprobable. Let us transform $M$ as follows. We contract the real axis to a convex envelope of the set comprising the set union $M$ and the point $A$, where the distance between $M$ and the point $A$ amounts to the minimum real number represented by computer. Let $M^*$ be union of $M$ and the above point. We consider instead of the map

$$ PR: M \rightarrow M $$

the following expansion of this map

$$ M^* \stackrel{i_1}{\rightarrow} R^P \stackrel{i_2}{\rightarrow} M^*, $$

where the map $i_1$ is present by the program and $i_2$ behaves so that, if its image is included into $M$, the map will not change; if it lies outside $M$, it will map to the chosen point $A$. Thus, instead of the map $M \rightarrow M$ that of $M^* \rightarrow M^*$ will be considered. When considering the following sequence of maps

$$ PR_1(M^*), PR_1(PR_1(M^*)),..., PR_1^{i_1}(M^*), $$

$$ PR_2(M^*), PR_2(PR_2(M^*)),..., PR_2^{i_2}(M^*), $$

the number of the remaining points will not increase after each mapping, i.e. if the measure $\mu^*$, $\mu^*(M^*) = 1$ is introduced, then

$$ \mu^*(PR_1^j(M^*)) \geq \mu^*(PR_1^{i+1}(M^*)) \text{ for all } i \geq 1, \text{ where } j=1, 2. $$

Since $M^*$ is finite, the numbers $J_1$ and $J_2$ will be available, hence

$$ \mu_1 = \mu^*(PR_1^j(M^*)) = \mu^*(PR_1^{i_1}(M^*)) \text{ for } jj \geq J_1, $$

$$ \mu_2 = \mu^*(PR_2^j(M^*)) = \mu^*(PR_2^{i_2}(M^*)) \text{ for } jj \geq J_2. $$

The numbers $J_1$, $J_2$ and $\mu_1$, $\mu_2$ are characteristics of the computers, $C_1$ and $C_2$ being not characteristics of the algorithm or of the program (the algorithm and the program are the same). Differences between $J_1$ and $J_2$ or between $\mu_1$ and $\mu_2$ may be particularly explained by the fact the digit network and the maximum real number represented by computers are not the same. As to $\mu_1$ and $\mu_2$ its practical evaluation is not easy because of a drastic consumption of machine time. The values $J_1$ and $J_2$ are more easy to be evaluated since the programs are the same and if the same input data are taken, the values $J_1$ and $J_2$ may be obtained. So, $J_1$ and $J_2$ or $\mu_1$ and $\mu_2$ may be considered as quality indices of hardware from the point of view of the programming itself (the programs and the input data...
are the same). If \( \mu_1 > \mu_2 \) (or \( J_1 > J_2 \)), it means that the capacities of the computer \( C_1 \) are greater than those of the computer \( C_2 \).

As an example, the program of computation has been used related to the root of the \( n \)-power from the complex number \( K \) [12]. The calculations have been made by means of the computers \( C_1 \) and \( C_2 \). The above calculations have shown that in the case of \( K=8 \), \( J_1 \) is equal to 26 for \( n=3 \) being equal to 13 for \( n=10 \) (computer \( C_1 \)). For computer \( C_2 \), \( J_2 \) is equal to 34 and to 17, respectively. Thus, \( J_1 < J_2 \), i.e. the capacities (complexity) of the computer \( C_2 \) are greater than those of the computer \( C_1 \).

3. Conclusion
The above approach permits to evaluate separately the hard- and software quality. It is obvious that, from the point of view of the theory of dynamic systems, approaches are marked as related to the separate assessment of the hardware and software quality. To the assessment of the software quality, such indices may be related as the information entropy \( IE \) and the fractal dimension \( d \). To the assessment of the hardware quality \( J_1 \) (\( J_2 \)) and \( \mu_1 \) (\( \mu_2 \)) may be referred. Thus, \( IE \) and \( d \) represent a quality index of software being \( J_1 \) (\( J_2 \)) and \( \mu_1 \) (\( \mu_2 \)) - a quality index of hardware.

So, such an approach from the point of view of the theory of dynamic system lets basically separate the quality indices relating them to those of software and hardware, as well as to the mixed ones, i.e. to such indices which are the same time of hard- and software. For instance, reliability may be related to the mixed indices [7]. The above approach to programming from the point of view of the theory of dynamic systems allows to see the ways of solving of such important indices as the quality indices of hardware and software which are based on the fundamental concepts of the modern physics, and the classification of programs. In the future, disparalleling of the programs will be added to the above indices. All the stated above seems to permit soon to pass from consideration of the programming as an art [12] to its consideration as a science based on the fundamentals of the ergodic theory, theory of information and informatics.

References
[1] Boehm B, Brown J R, Kaspar H, Lipow M, MacLeod G J and Merritt M J 1978 Characteristics of Software Quality (Amsterdam: North-Holland)
[2] Boehm B 1981 Software Engineering Economics (New Jersey: Prentice Hall)
[3] Perry W E 1986 Hatching the Data processing Quality Assurance Function (Orlando: Prentice Hall)
[4] Pyle I C 1991 Developing Safety Systems (New York: Prentice Hall)
[5] Kopyltsov A V and Boiko A V 1993 Proc. Int. Conf. on Informational Technology and People (Moscow) Vol 2 (Moscow: Moscow State University Press) p 121
[6] Kopyltsov A V 1993 Proc. Int. Conf. on CAD\( \times \)CAM, Robotics and Factories of the Future (St.Petersburg) Vol 2 (St.Petersburg: Nauka Press) p 539
[7] Kopyltsov A V and Vorobiev V I 1993 Proc. Int. Conf. on Computer Systems and Applied Mathematics (St.Petersburg) Vol 1 (St.Petersburg: Nauka Press) p 178
[8] Kan S H 2002 Metrics and Models in Software Quality Engineering (Boston: Addison-Wesley Longman Publishing Co.)
[9] Wagner S 2013 Software Product Quality Control (New York: Springer)
[10] Suryanarayana G 2015 IEEE Software 32 7
[11] Nicolis G and Prigogin I 1989 Exploring Complexity (New York: W.H. Freeman and Company)
[12] Knuth D E 1969 The Art of Computer Programming (Massachusetts: Addison-Wesley Publishing Co.)
[13] Billingsley P 1965 Ergodic Theory and Information (New York: John Wiley and Sons)
[14] Cornfeld I P, Sinai Y G and Fomin S V 1980 Ergodic Theory (Moscow: Nauka Press)
[15] Moon F C 1987 Chaotic Vibrations (New York: John Wiley and Sons)