The α Metric Values For Random and Fully Organized Programs

Ana Isabel Cardoso
Eng.de Sist. e Computadores
Universidade da Madeira
Tel:+351-291740320
aic@dragoeiro.uma.pt

Rui Gustavo Crespo
Instituto Superior Técnico
Univ. Técnica de Lisboa
Tel:+351-218418398
R.G.Crespo@digitais.ist.utl.pt

Peter Kokol
Lab. for System Design
University of Maribor
Maribor,Slovenia
kokol@uni-mb.si

Abstract

Software products are complicated and complex systems. Thereafter, the conventional engineering techniques for quality measure are not good enough. The methodologies suitable for the analysis of complex systems, namely "the theory of chaos", can be used for the identification of the program organizations.
In this paper we introduce a method to assess the complexity of source and object programs. We analyse how the new fractal software metric, called α, shows the presence of the semantic structure in programs and may quantify their complexity.

1. Physically-based View of Software

The analysis of many physical systems in the past, such as the behaviour of fluid systems, raised the creation of theory of chaos, now named as the complex system theory. Nowadays, the complex system theory has a broadband application in areas, such as economics [2], population growth [3] and linguistics[6].
Complex systems, such as the atmosphere dynamics, although reveal pseudo-random behaviour, obey to general laws. The elements of complex systems are large, with many degrees of freedom, and the behaviour of each element depends on all the other elements and on the system history. For example, the weather forecast depends on the physical elements, such as pressure and temperature, of every point of the atmosphere, and their interaction in the previous periods [9]. This behaviour makes the long-term prediction to become very difficult to obtain. Yet, the complex systems reveal generic emergent trends resulting from the balance of all element interactions. For example, in the Mediterranean regions, the temperatures in the summer are higher than in the winter.
We advocate that the software development and products are complex, dynamic, non-linear and adaptive. Computer programs, including information systems, usually consist of number of entities like subroutines, modules, functions, etc., on different hierarchical levels [4,11].

The laws for software process development have been identified informally long time ago [8].
The first law says that large systems are never completed; they continue to evolve till they became useless. The result of this evolution is the large number of software releases. For example, the Linux operative system, distributed by Red Hat, hold release number 8 by the end of 2002 and many patches have been made available in between.
The second law tells us that as software systems evolve, the complexity grows unless project managers take actions to reduce the complexity. So, the control of complexity is an important issue in software engineering.
Conventional methods for measuring software products, unfortunately, have not been successful enough. Then, we can gain fundamental understanding of the software design process and products, using the science of complexity, theory of chaos, fractals and other not yet tried "physically based" methodologies.

2. How to measure the structure complexity in a program?

To use the analysis techniques of complex systems, we need to transform a program into a "signal''. Programs are always string files, with the basic alphabet equal to characters in the source files and binary digits in the object files. Whatever representation form is, programs can always be coded as a sequence of numeric codes. For the source programs, we are concerned only to codification the reserved words, because those are the only to express actions.

The correlation metrics measure the connection between the same symbol elements in different string positions. The short-range correlation, such as verified in the Markov chains, relates elements positioned very closed. The long-range correlation LRC [7,10], named α metrics, filters short-range fluctuations and reveal the basic structure of the string. Therefore, the α metric is the most natural candidate for the assessment of the program structures.

When the correlation length, defined as the maximal propagation distance of the effect of some disturbance within a system, increases, the small variations do not disappear. In this case, the small variations merely become a finer structure, superimposed to the large-scale structure. As a consequence, the LRC is very important for understanding the system's behaviour, since we can quantify it with one exponential rate [5].

There are several methods to evaluate the long-range correlation. We selected the commonly Brownian cumulative random walk, also known as Peng's method [12], and identify the characteristic function F(l) as the root of mean square fluctuation about the average of the displacement.

The *F(l)* can distinguish two possible types of behaviour:

- if the string sequence is uncorrelated (normal random walk) or there are local correlations extending up to a characteristic range i.e Markov chains or symbolic sequences generated by regular grammars, then

  $$F(l) \approx l^{0.5}$$

- if there is no characteristic length and the correlations are "infinite" then the scaling property of *F(l)* is described by a power law

  $$F(l) \approx l^{\alpha} \text{ and } \alpha \neq 0.5.$$

Our goal is to verify how the long-range correlation is linked to the program structure complexity. For that, for each member of a program sample, we compare it with a sequence of generated programs. The sequence starts from a random program and the other members of the sequence adds, step-by-step, organizational information.

Each sample program is done consciously and with purpose. Because the purpose imposes organizational structures, we anticipate that long rang metric should differ from 0,5. The presence of a long-range correlation different of 0,5 indicates the presence of scaling proprieties. So, it indicates that there is a structure that repeats itself in all scales.

Using the same hypothesis, the more information is added to the random program, the closer to the initial value the α should be.

3.A Case Study of α Values

The sample is a set of 36 programs with the same goal, the generation of a compiler for the same language [1]. The sizes of source files are greater than 2000 Lines of Code. They are all syntactically correct, but the implemented functionalities vary. The project teams have similar knowledge of compiler technology and the same programming practices.

The programs, without purpose, are randomly generated from each sample program and have the same frequency distribution for the reserved words.

3.1 Measuring Source Files

The numeric codes for the reserved words are centred in 0. For the coded string of reserved words, we calculate the α value using the Peng's method.

The figure 1 represents the long range correlation for a set of compilers codified in C language, as the series 1. The average value for α is 0.82±0.06 for the compiler programs. This value represents a high level of structures, since the maximum value for α is 1.

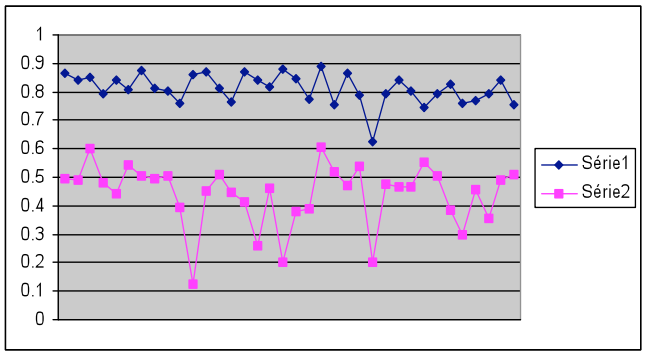

Fig.1 Long rang correlation for two different sets of programs, calculated on source files

For the compilers, we also verified that, the α values are independent of the number of implemented functionalities
The compiler source programs are divided in two parts: code-generated part, for the lexical and parser parts, and user-defined code. The α value is identified form the concatenation of all source files, in the order specified by the *link* command. The figure 2 shows that, changing the order in the concatenation of the source files, does not statistically affect the α values

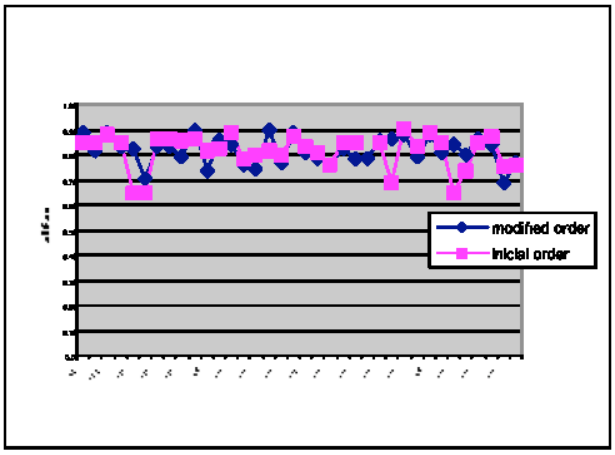

Fig.2 Long rang correlation for two different concatenation orders

3.1.1 Random Programs With Equal Frequencies

The figure 1 represents, in the series 2, the long range correlation for random programs, which are nor syntactically correct, but have equal distribution of reserved words. The average value for α is 0.48±0.02.
We conclude that, the LCR metrics distinguish programs with meaning from those without meaning.

3.1.2 Syntactically Correct Random Programs

We increased the organization of the random programs having equal distribution of reserved words (depicted as the series 2 of figure 1), with syntactic correctness. The number of functions is very small, usually less than two. The average value for α raised to 0.58±0.04.
We conclude that, the α metrics is influenced by the syntactical correctness. Because the α values are still away from those of the source codes, this influence is small are there are other factors more important.

3.1.3 Random Programs With Same Number of Functions

We provided a further level of organization for the programs depicted in the section 3.1.2, with equal number of functions. The plot for the α values for the sources and these random programs are depicted in the figure 3. The average value for α becomes 0.81±0.12. When the number of functions becomes much greater than the number of functions in the source files, the α values decreased again.
We conclude that, the α metric values measure the semantic organization revealed by the equal distribution of reserved words and the same number of functions.

The α metrics do not differentiate the file specifications, because we also obtained the same high values for programs, derived from different specifications and having different distribution of reserved words and different number of functions.

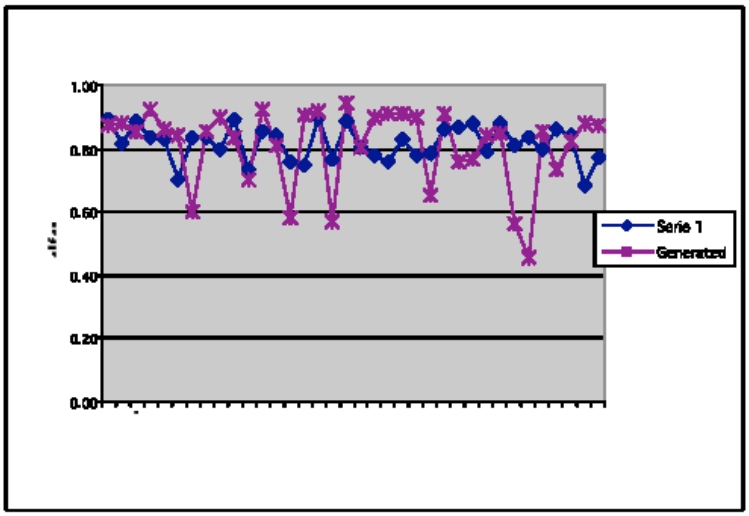

Fig.3 Long rang correlation for generated programs with same distribution and functions number

3.2 Measuring Object Files

The Peng's method may also be applied to the object files, because they are a string of 0's and 1's. The average value for α is 0.72±0.08. The measures calculated on source and object programmes are strongly related, as depicted in the figure 4, with correlation coefficient equal to 0.82

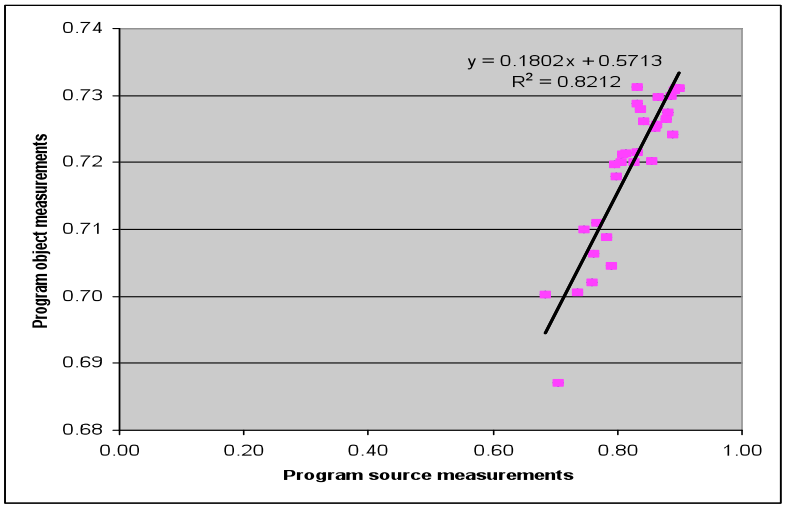

Fig.4 correlation between the source and object file α values

The strong correlation between source and object files, enable project managers to select the file they wish. Moreover, the file limitation problems may be partially overcomed. When the source files are small, we can calculate the α values on the object files.

There is an interest to identify the α values for each of the source files and compare them to the α value for the all system. We identified a greater variation in the α values, for each user object file, in relation to the α value for the system. This suggests that the organization is increased at the system level, but more studies must confirm this hippothesis.

Conclusions

The long rang correlation may be calculated for computer programs, both at source and object files, and provides a basis for better understanding the organization of computer semantic structures.

References

[1] Aho, Setti, Ullman, Compiler Principles, Techniques and Tools, Addison-Wesley, 1986

[2] Artur, W.B. et al.; The Economy As An Evolving Complex Systems II, Santa Fe Institute, 1997

[3] Buldyrev,S.V. et al.; Fractals in Biology and Medicine, Springer-Verlay, 1994

[4] Cohen B, Harwood W T, Jackson M I: *The specification of complex systems*, Addison Wesley, 1986.

[5] Crespo,R.G, Cardoso,A.I. and Kokol,P., Complexity-based Metrics for the Evaluation of the Program Organization, in-printing

[6] Kokl, P., Computer and Natural Language Texts- a comparison based on long-range correlations, Journal of American Society for Information Science, pp 1295-1301, 1999

[7] Kokol, P., Brest, J., Žumer, V., "Long-range correlations in computer programme", Cybernetics and systems 28(1), pp. 43-57, 1997.

[8] Lehman,M.M., "Programs. Life cycles and the laws of software evolution", Proc. IEEE, 15(3), pp. 225-252, 1980.

[9] Peitgen, R et al. Chaos and Fractals - New Frontiers of Science, Springer Verlag. 1993

[11] Watt, D. A.: *Programming Language Concepts and Paradigms*, Prentice Hall, 1990.

[10] Schenkel, A., Zhang, J., Zhang, Y (1993). Long range correlations in human writings, *Fractals* 1(1),47-55.