UDC 681.3.068

V. MISHCHENKO

V.N. Karazin Kharkov National University, Ukraine

ONE EXPERIMENT IN USING ENERGY METRICS PROPOSED FOR SOFTWARE PROCESS ASSESSMENT

According to the standard IEEE 982, the functions of a measure of software product/process can be provided by Halstead's so called Software Science Measures. We had built a new approach in this area earlier, so that "energy of specifications" and "work of coding" arose in the new theory that is inherited the old Software Science. In particular we obtained theoretical ability to test reliability of software development. Practical using aspects of the new measure are investigated on example of development of a real software product. We select one AdaCore product which is distributed under open source license. Our example of calculation the metric uses sequence of sources that was produced on different stages of this product development.

software process quality, Software Science, metric, Ada language, tokens, program structure, language level

The problem

Standard ISO/IEC 9126 [1, 10] requires, roughly say, that the software process can be adequately represented by two phases, which, if necessary, may be repeated. The first phase includes such early stages, as requirements definition, design specification and source code. At this phase the quality of current state of the future software product can be estimated in the form of internal quality of this product. The second phase starts, when developing software reach possibility to be tested as a part of the target system (or its simulator). Then the quality of developing software product can be evaluated in the form of external quality. However, sometimes this model may not be enough adequate. Possibly in that occasion it would be useful to represent software process by sequence of product's versions. Can we assess the reliability of the software product by means of evaluating its development process? It leads us to the standard IEEE 982 [1, 11], which provides us conditions of positive answer. Next, can we assess the reliability of such process by means of measuring attributes, which reflect changes of source code and documentation?

We had proposed an answer on this actual question at [2]. Note that our approach was based on development of Software Science Measures [2 – 4]. The IEEE 982 [1, 11] allows us to apply them to measuring quality of software products and software processes. The new measure, named "intellectual heat" [2], is applicable to qualities of bough products and processes too [2, 5]. However the only offer of use a new metric can not introduce this metric in practice. That is strongly desirable to check such metric's properties as reliability, correctness, meaningfulness, cost effectiveness, availability, and indicativeness (ISO/IEC 9126.3,A.2.1 [10]).

In this paper we present new results of testing the mentioned metric to check five mentioned properties in relation to applications in the field of scanning analysis of formal texts. To simplify our task we restricted ourselves with applications which code is written in Ada language. In that case we could calculate needed attributes with source code only (without documentation). It is possible so any Ada program is self-documented because of all Ada program units shall be specified concerning their interfaces by means of Ada language oneself [6]. Earlier elementary problems of Software Science metrics for Ada software had been analyzed in [7].

New metric and the task statement

We shall consider the next attributes [2] of sources:

E – specification energy;

A – work of coding (meaning, it is “in ideal case”);

$Q = E - A$ (informational heat of software development) or alternatively

$$q = Q/\max(E, A) \qquad (1)$$

– normalized informational heat currency.

E , A are different generalizations [8] of Halstead’s measure for programming efforts, which defined by him for simple subprograms only [4].

We define the metric, which would be named “Energy compliance”. Purpose of the metric is question “How compliant are the actual directions of informational heat to the required directions?” Its attribute shall be evaluated for sequence of product’s versions S_1 (earliest), .. S_n (latest). Let $q_i = q(S_i)$. The formula is

$$X = (C_1 + ..C_n)/n \qquad (-1 \le X \le 1), \qquad (2)$$

where $C_i = 1$ if q_i complies general description of S_i, $C_i = -1$ if not complies, and $C_i = 0$ if is vague.

General description (GD) for S_i is manager’s prediction (or conclusion): what should be focusing the work on this version? Either is it design or coding?

To evaluate (2) for a versions we shall define the reliability level $\alpha \in (0;1)$. Then we must extract the formal interpretation from each general description. Alternative interpretations are “The release focuses on the design of the structure” ($\alpha < q_i$ is complied, $-\alpha/2 < q_i \le \alpha$ is vague), “Work focuses on the development of the code” ($q_i < -\alpha$ is complied, $-\alpha \le q_i < \alpha/2$ is vague), “Feature is the balance efforts to design and coding.” ($|q_i| \le \alpha/2$ is complied, $\alpha/2 < q_i \le \alpha$ is vague). So our metric is meaningful enough.

Our task is to investigate others of required properties of this metric, relying both on the logical considerations and on the experiment on the real example.

Minimizing subjective factors

Most of the properties of the metric (2) depend on objectivity of measurement needing attributes.

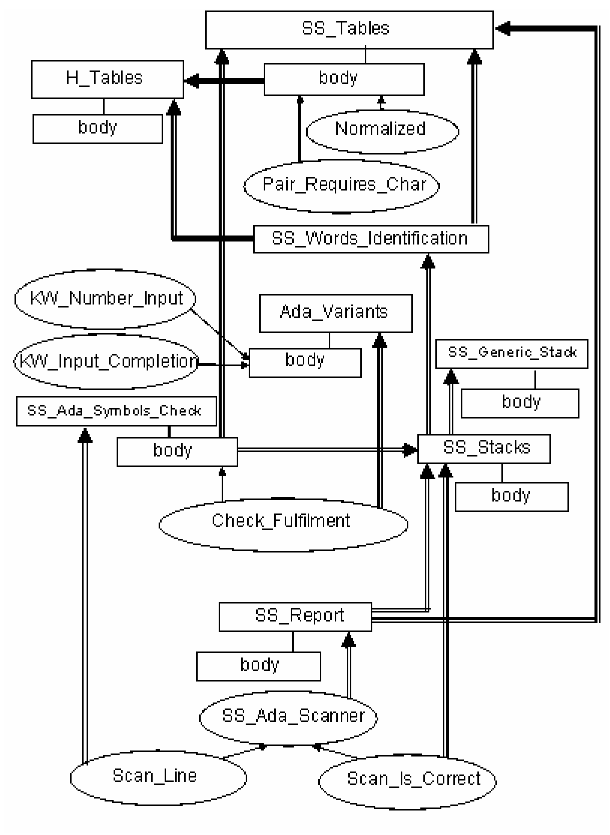

Fig. 1. The scheme of program structure (SPS) for *SS_Ada_Scanner* that help us count tokens of programs

We could have two sources of human misunderstanding. First is erroneous GD (general description), and second is inaccuracy of them formal interpretation. Therefore, preparing applications of this metric, it is necessary to enforce methodological recommendations.

Table 1

Classification of “tokens of programs” for Ada language

Kind numbers	Description	Examples
1- 11	11 from 16 Ada delimeters	& >
12 - 18	7 from 10 compound delimeters	.. <>
19	All programmer’s identifiers	My_Prog
20-53	39 from 64 reserved words	abs
54-57	Different forms of number	-47.0
58-61	Characters, strings, comments	‘a’ “Aa”--
62-69	Others simple or compound delimiters, their groups	( ) << >>
70-113	Others reserved words, their groups	abort with abort

Now we shall assume that source code under consideration is written in Ada. The first technical problem is description of “counting strategy” (see, for example, [7]). In order to overcome it, we have a precise definition of all possible types of Halstead's "tokens of programs" for Ada 95 language (see table 1). However, we make no distinction between operators and operands [9].

Because of context dependences the scanner of tokens is based on the recognizer with stacks. The scanner structure is shown on the fig. 1 (the packages are shown as rectangles, subprograms as ovals, subunit dependences as "$\to$", context dependences as "$\Rightarrow$").

The second factor of possible unreliability is our metric relates to the calculation of specification energy:

$$E = \sum_i \sum_k E\left(G_i^k\right) \qquad (G_i^k = \left\{B_i^{k,j}\right\}_i), \tag{3}$$

where i – index of library unit L_i, that is any (generic) package declaration or any subprogram body;

G_i^k – group of so called SPS-blocks $B_i^{k,j}$ of L_i, which are subprograms, tasks without entries, or entries.

There are three strong signs that the blocks belong to a particular group. The first sign is that they are included in the task, protected unit, or package. The second is that they are operators under the same type. The third is that a group is pointed by suitable commentary. Otherwise the system of groups, and, thereby, the importance of measure X (see (1) – (3)) are under doubts.

The last of technical problems relates to the fact that our computing method is based on potential vocabulary η^* [4, 2], it is not as in [11]. The η^* depends on η_2^* as in [4], but we have defined it strongly:

$$\eta_2^* = p + \sqrt{j_1 \cdot j_2}, \tag{4}$$

where η_2^* – number of potential operands of a SPS-block [2]; p – number of all formal parameters of this block; j_1 – number of files are accessible in the body of this block; j_2 – total number of input-output operators.

We may conclude that we have justified the properties of the metric: reliability, availability, correctness.

First test of effectiveness

Here we consider the first experience in test cost effectiveness of our metric. That is cost-benefit relation.

We tried to answer question "Will we obtain more important result if we apply more expense?"[10]. The analysis object was AdaCore tool *gnatpp* [12]. On beginning of 2007 it had 45 compilation units, which program volumes [4, 11] were distributed as is in table 2, were volumes measured in unit named "Halstead" [8],

$$1\,Hd = 1000\,bit \times token \tag{5}$$

Table 2
Number of *gnatpp* units in different ranges of volume

Range	0..1	1..4	4..16	16..32	32..64
Number	17	11	7	7	3

The attribute A, which is needed for q according to (1), depends on volumes of all compilation units. In case of *gnatpp* our *SS_Ada_Scanner* calculates these volumes for 5-10 min including timeouts to check each calculated value visually. However attributes A and E both are calculated on base of analysis all SPS-blocks too. The *gnatpp* source from 2007-01 contains approximately 500 such blocks. 228 of them require careful human attention to union them to round 20 groups. The "handle" analysis occupies around one work day. Our experience allows us to design such tool which will shorten that time to one hour. Anyway, it seems essential expenses. Let us first present *gnatpp* process by means the pair of versions: initial (2001-06) and final (2007-01). Since $q_{init} > 0{,}5,\ q_{fin} < -0{,}5$, we conclude that the first is designed structure of the software, and later its code. In such cases, 1.0 is the most likely value of the attribute (2), and it is not informative. A much more important result would be to determine when the development has reached the balance? This must consider intermediate versions of the product (e.g. three, see table 3). In doing so, we have sought that the turning point was 2002 (at any level of reliability > 0,12). Now it is informative. Indeed, if any link between the GD and q is absent, the probability of such a value may be 0,17, and even with a reasonable guess it may be only 0,25.

If we examine more intermediate releases, probability of $X \approx 1$ (2) with arbitrary descriptions will be even less. So do not doubt the efficiency of additional costs.

Table 3

Profile of metric Energy Compliance for rel.-s of *gnatpp*

r.	init	2001-09	2001-12	2003-01	fin
q	0,71	0,40	0,12	-0,57	-0,58

Note, in calculating energy (3) with the help of formulas [4, 2], we used an indicative value 1.5 of the Ada language level λ. At the same time, our study provides statistics useful to clarify this value in the future.

Table 4

Number of *gnatpp* library units in different ranges of λ

Range	0..0,57	0,57..1,7	1,7..5	5 ..15	15.. ∞
Number	14	3	4	1	1

Conclusion

The new software process metric has been defined. It uses modernized Hallstead's measures as primitives. It is of the subcategory Management Control, though old Hallstead's process metrics belong to another.

In relation to this metric, we had explored properties of reliability, availability, cost effectiveness, correctness, and meaningfulness. Efforts of the metric calculation had been evaluated on the real sample. Regarding Ada units of a real software project, the language level had been obtained in the form of heuristic distribution.

It would be desirable to test indicativeness [10] of this metric.

References

1. Kharchenko V.S., Sklyar V.V., Tarasyuk O.M. Methods of modeling and estimation of quality and reliability of the software. – Manual. – Kharkov: National aerospace university "KhAI, 2004. – 159 p. [in Russian].

2. Mishchenko V.O. Mathematical model of style Software Science for the metric analysis of complex scientific programs // Bulletin of V. Karazin Kharkiv National University. – 2004. – №629. Series "Mathematical Modeling. Informational Technologies. Automated Control Systems" Issue 3. – P. 70-85 [in Russian].

3. Mishchenko V.O. Software of DSM: a role of mathematical models of reliability and labor input // Proceedings of International school-seminars «DSMM-Ph», Oryol, OSU, 2006. – Vol. 4. – P. 73-80 [in Russian].

4. Halstead M.H. Elements of Software Science / Translation from Englesh to Russian by Yufa V. – M.: Finance and Statistics, 1981. – 128 p.

5. Gahov A. Testing a new approach to the analysis of projects development using generalization parameters offered by software science // Transactions of International Conference SCALNET'04, 28-30 September 2004. – P. 118-120.

6. John Barnes. Programming in Ada 2005. – Addison-Wesley. Pearson Education. – 2006. – 828 p.

7. Miller D.M., Maness R.S., Howatt J.W., Shaw W.H. A software science counting strategy for the full Ada language //ACM SIGPLAN Notices, May 1987, Volume 22 Issue 5, ISSN:0362-1340. – P. 32-41.

8. Mishchenko V.O. The application of mathematical modeling to system analysis of software supplied the method of discrete singularities // Transactions of VII International Symposium "Discrete Singularities Methods in Mathematical Physics"– Theodosia, 1997. – P. 117-120 [in Russian].

9. Mishchenko V.O. Improvement of Halstead's mathematical model and its development to assess metrical characteristics of modern software // Mathematical Modeling. Scient.Transactions / NAS Ukraine, In-t of Mathematics. – Kiev, 1996. – P. 180-181.

10. "ISO/IEC TR 9126 3:2003". – [Електр. ресурс]. – Режим доступу: http://www.iso.org/iso/en/ CatalogueDetailPage.CatalogueDetail?CSNUMBER=22891.

11. "982.1" (IEEE Computer Society Document). – [Електр. ресурс]. – Режим доступу: http: //members.aol.com/geshome/IEEE982/IEEE9821.pdf.

12. "ASIS". – [Електр. ресурс]. – Режим доступу: http://www.adacore.com/home/gnatpro/add-on_techno logies/asis.

Надійшла до редакції 5.04.2007

Рецензент: д-р техн. наук, проф. В.С. Харченко, Національний аерокосмічний університет ім. М.Є. Жуковського «ХАІ», Харків.