A new approach to defining software complexity measures

Z. DARÓCZY, L. VARGA

Abstract

A general method is given for defining software complexity measures. Properties of the complexity measure are given by functional equation. Three cases of functional equations are discussed. Many known software complexity measures are given as special cases of the solutions of functional equations and a new measure is also presented.

Introduction

It is a fact of common knowledge, that both simple and complicated programs can be developed for the solution of a given problem, whatever its inherent complexity. Therefore software complexity can be investigated independently of the complexity of a problem. During recent years many efforts have been made to create quantifiable measures of software complexity ([1], [3]), to use objective complexity measures in programming methodology and to validate these uses with empirical researches [2].

In spite of the importance of software complexity it is insufficiently known and defined. In this paper a new approach to defining abstract properties of software complexity over the class of structured programs is proposed by the help of using functional equations. Many known measures of software complexity can be obtained as special cases of the solutions of functional equations, and a new measure is also presented.

Software complexity measure

Given a system (X, F, A) where X is a set of data, F is a set of functions $(f: X \to X)$ and A is a set of predicates $(a: X \to \text{Bool})$.

Definition 1. Simple programs are:

1. null, with program function x,
2. assignment f, with program function $f(x)$, for all $f \in F$.

Let SP be the set of all simple programs.

Definition 2. Structured programs are:

1. Simple programs.
2. Sequence $B(u, v)$ with the program function $p_v(p_u(x))$;
3. Selection I—T—E $(a; u, v)$ with the program function **if** $a(x)$ **then** $p_u(x)$ **else** $p_v(x)$;
4. Repetition W—D$(a; u)$ with the program function $p(x)=$**if** $a(x)$ **then** $p(p_u(x))$ **else** x;

where $a \in A$, u and v are structured programs with program functions $p_u(x)$, $p_v(x)$ respectively.

Let S be the set of all structured programs.

Given the complexity measures

$$b\colon \mathrm{SP} \to N,$$

$$c\colon A \to N$$

and the homomorphisms

$$g\colon N^2 \to N,$$

$$h\colon N^3 \to N,$$

$$j\colon N^2 \to N.$$

Definition 3. Complexity measures of $B(s_1, s_2)$, I—T—E $(a; s_1, s_2)$, W—D $(a; s)$ are:

$$b(B(s_1, s_2)) = g(b(s_1), b(s_2)),$$

$$b(\text{I—T—E}(a;\ s_1, s_2)) = h(c(a), b(s_1), b(s_2)),$$

$$b(\text{W—D}(a;\ s)) = j(c(a),\ b(s).$$

The question is what kind of functions g, h, j characterize the software complexity measure of structured programs sufficiently? In order to find an appropriate complexity measure the properties of functions g, h, j will be given by functional equations.

First approximation. If the functional equations

$$g(x+x', y+y') = g(x, y)+g(x', y'),$$

$$h(x+x', y+y', z+z') = h(x, y, z)+h(x', y', z'),$$

$$j(x+x', y+y') = j(x, y)+j(x', y'),$$

hold, then the functions g, h, j give an *acceptable measure* for each structured program.

Theorem 1.

$$g(x, y) = c_1x + c_2y,$$
$$h(x, y, z) = d_1x + d_2y + d_3z,$$
$$j(x, y) = e_1x + e_2y$$

where $c_1, c_2, d_1, d_2, d_3, e_1, e_2$ are integer constants.

Proof.

$$x' = y = 0 \Rightarrow g(x, y') = g(x, 0) + g(0, y'),$$
$$x = x' = y = y' = 0 \Rightarrow \quad g(0, 0) = 0,$$
$$y = y' = 0 \Rightarrow g(x+x', 0) = g(x, 0) + g(x', 0).$$

This is the well known Chauchy equation, which has the following solution:

$$g(x, 0) = c_1x.$$

Similarly we have

$$x = x' = 0 \Rightarrow g(0, y+y') = g(0, y) + g(0, y') \Rightarrow g(0, y) = c_2y.$$

That is, $g(x, y) = c_1x + c_2y$.

Second approximation. If

$$g(x+x', y+y') = g(x, y) + g(x', y'),$$
$$h(x+x', y+y', z+z') = h(x, y, z) + h(x, y', z') + h(x', y, z) + h(x', y', z'),$$
$$j(x+x', y+y') = j(x, y) + j(x, y') + j(x', y) + j(x', y'),$$

then g, h, j give an *adequate measure* for each structured program.

Theorem 2.

$$g(x, y) = c_1x + c_2y,$$
$$h(x, y, z) = x(d_1y + d_2z)$$
$$j(x, y) = exy$$

Proof.

$$x = x' = y = y' = 0 \Rightarrow j(0, 0) = 0$$
$$x' = y = y' = 0 \Rightarrow \quad j(x, 0) = 0$$
$$x = x' = y' = 0 \Rightarrow \quad j(0, y) = 0$$
$$y' = 0 \Rightarrow j(x+x', y) = j(x, y) + j(x', y) \Rightarrow j(x, y) = e(y)x$$
$$x' = 0 \wedge x \neq 0 \Rightarrow e(y+y') = e(y) + e(y') \Rightarrow e \cdot y$$

That is

$$j(x, y) = exy.$$

Similarly we have

$$h(x, y, z) = x(d_1 y + d_2 z).$$

Third approximation. If

$$g(x+x', y+y') = g(x, y)+g(x', y')$$
$$h(x+x', y+y', z+z') = g(x+x', y, z)+g(x+x', y', z')$$
$$j(x, y) = h(x, y, l)$$

then g, h, j give a *correct measure* for each structured program.

Theorem 3.

$$g(x, y) = c_1 x_1 + x_2 y$$
$$h(x, y, z) = d_1(x)y + d_2(x)z$$
$$j(x, y) = d_1(x)y + d_2(x)$$

where $d_1(x)$, $d_2(x)$ are unknown functions.

Proof.

$$x' = 0 \Rightarrow g(x, y+y', z+z') = g(x, y, z)+g(x, y', z') \Rightarrow$$
$$\Rightarrow g(x, y, z) = d_1(x)y + d_2(x)z$$

Special cases

1. *First approximation*

$$g(x, y) = x+y$$
$$h(x, y, z) = x+y+z$$
$$j(x, y) = x+y$$

1.1. Let $b(f_i)=b_i, \quad f_i \in F;$

$$c(a_i) = c_i, \quad a_i \in A.$$

Then

$$b(s) = \sum_{i=1}^{\varphi} b_i + \sum_{i=1}^{\pi} c_i$$

where $s \in S$ and

π=number of predicates in s;
φ=number of functions in s.

1.2. Let $b_i=c_i=1$ for $i=1, 2, \ldots$, then

$$b(s) = \varphi+\pi.$$

1.3. Let $c_i=1$ and $b_i=0$ for $i=1, 2, \ldots$, then $b(s)=\pi$,

which gives the well known McCabe metric [1].

2. *Second approximation*

$$g(x, y) = x+y$$

$$h(x, y, z) = x(y+z)$$

$$j(x, y) = xy$$

2.1. Let $b(f_i)=b_i, \quad f_i \in F,$

$$c(a_i) = c_i, \quad a_i \in A,$$

then we get the Prather measure [3].

3. *Third approximation*

$$g(x, y) = x+y$$

$$h(x, y, z) = d_1(x)y+d_2(x)z$$

$$j(x, y) = d_1(x)y+d_2(x)$$

3.1. Let $d_i(c(a_j))=d_{ij}$; $i=1, 2$; $a_j \in A$, where
d_{1j}=number of "true" value in the operation table of predicate a_j;
d_{2j}=number of "false" value in the operation table of predicate a_j
and $b(f_i)=b_i$, $f_i \in F$ which produces a new measure.

An example

Let

$$s\text{: } \textbf{if } a_1 \textbf{ then } s_1\text{; } \textbf{while } a_2 \textbf{ do } s_2 \textbf{ od}$$
$$\textbf{else } s_2 \textbf{ fi; } s_4\text{;}$$

and

$$c(a_i) = c_i, \quad b(s_i) = b_i; \quad d_i(c(a_j)) = d_{ij}.$$

The complexity measures in the above special cases are:

1.1.: $c_1+c_2+c_2+b_1+b_2+b_3$

1.2.: 6

1.3.: 2 (McCabe)

2.1.: $c_1b_1+c_1c_2b_2+c_1b_3+b_4$ (Prather)

3.1.: $d_{11}b_1+d_{12}d_{21}b_2+d_{12}b_3+b_4+d_{11}d_{22}$

L. KOSSUTH UNIVERSITY
DEBRECEN
HUNGARY

L. EÖTVÖS UNIVERSITY
BUDAPEST
HUNGARY

References

[1] T. J. McCabe, A Complexity measure, IEEE Trans. Software Eng. 2. (1976) 308—320
[2] B. Curtis, In search of software complexity, Workshop on Quantitative Software Models for Reliability, Complexity, and Cost. New York, IEEE (1980).
[3] Ronald E. Prather, An axiomatic theory of software complexity measure, The Computer Journal 4 (1984) 340—347.

(Received Jan. 30, 1987)