Oriental Renaissance: Innovative, educational, natural and social sciences
Scientific Journal Impact Factor
Advanced Sciences Index Factor

VOLUME 2 | ISSUE 4/2
ISSN 2181-1784
SJIF 2022: 5.947
ASI Factor = 1.7

DASTURIY TA'MINOT SIFATINI BAHOLASHDA HALSTEDNING MURAKKABLIK O'LCHOVLARIDAN FOYDALANISH

Sharipov Bahodir Akilovich,
Toshkent Axborot Texnologiyalari Universiteti, "Tizimli va amaliy dasturlashtirish" kafedrasi katta o'qituvchisi
bahodir.61@mail.ru

ANNOTATSIYA

Ushbu maqolada dasturiy ta'minotning sifatini baholashda qo'llaniladigan miqdoriy usullardan biri Halstedning murakkablik o'lchovlari ko'rsatib berilgan. Halstedning murakkablik o'lchovlarini yoritib berish uchun misol sifatida "Pufakchali saralash algoritmi"ning C++ tilidagi dasturidan foydalanilgan. Dasturning metrik xarakteristikalarini hisoblash uchun Halsted tomonidan taqdim etilgan murakkablik o'lchovlariga doir misollar keltirilgan.

Kalit so'zlar: *murakkablik, o'lchov, dastur, operator, operand, uzunlik, hajm, algoritm, energiya, proportsional, Kod, xato, samaradorlik, baholash, ko'rsatkich, vaqt.*

ABSTRACT

This article provides a measure of software complexity using the Halstead metric, one of the quantitative methods used to evaluate software quality. To highlight the Halsted metric, the bubble sort algorithm program in C++ is given as an example. Examples of complexity measurements provided by Halsted are given to calculate program metrics.

Keywords: *complexity, measurement, program, operator, operand, length, volume, algorithm, energy, proportional, code, error, efficiency, evaluation, indicator, time.*

АННОТАЦИЯ

В этой статье приведена измерения сложности программного обеспечения с помощью метрики Холстеда, один из количественных методов, используемых для оценки качества программного обеспечения. Для освещения метрики Холстеда в качестве примера приведена программа алгоритма пузырьковой сортировки в C++. Приведены примеры измерений сложности, предоставленные Холстедом, для расчета метрических характеристик программы.

Ключевые слова: *сложность, измерение, программа, оператор, операнд, длина, объем, алгоритм, энергия, пропорциональный, код, ошибка, эффективность, оценка, показатель, время.*

KIRISH

Dasturiy ta’minot ko‘rsatkichi - bu dasturiy ta’minotning ba’zi xususiyatlari yoki uning texnik xususiyatlarining raqamli qiymatini olish imkonini beruvchi o‘lchov. Miqdoriy usullar boshqa sohalarda yaxshi ishlaganligi sababli, ko‘plab kompyuter olimlari va amaliyotchilar ushbu yondashuvni dasturiy ta’minotni ishlab chiqishga o‘tkazishga harakat qilishdi. Tom DeMarko aytganidek, “siz o‘lchay olmaydigan narsani nazorat qila olmaysiz”.

Amaldagi o‘lchovlar to‘plamiga asosan quyidagilar kiradi:

– o‘sish tartibi (asimptotik tahlil va O-notatsiya nuqtai nazaridan algoritmlarni tahlil qilishni anglatadi),
– kod satrlari soni,
– siklomatik murakkablik,
– funktsiya nuqtalarini tahlil qilish,
– kodning 1000 satridagi xatolar soni,
– test orqali kodni qamrab olish darajasi,
– talablarni qoplash ,
– sinflar va interfeyslar soni,
– bog‘liqlik.

Avvalo, dasturlarning dastlabki kodining miqdoriy xususiyatlarini hisobga olish kerak. Eng elementar ko‘rsatkich bu kod satrlari soni (SLOC). Ushbu ko‘rsatkich dastlab loyihaning mehnat xarajatlarini baholash uchun ishlab chiqilgan. Biroq, bir xil funktsiyani bir nechta satrlarga bo‘lish yoki bitta satrda yozish mumkinligi sababli, bir qatorda bir nechta buyruqlar yozilishi mumkin bo‘lgan tillar paydo bo‘lishi bilan metrikani deyarli qo‘llamaydi. Shuning uchun kodning mantiqiy va jismoniy qatorlari o‘rtasida farqlanadi. Kodning mantiqiy qatorlari dastur ko‘rsatmalari sonidir. Ta’rifning ushbu versiyasi ham o‘zining kamchiliklariga ega, chunki u ishlatiladigan dasturlash tili va dasturlash uslubiga juda bog‘liq. Miqdoriy xarakteristikalarga SLOCdan tashqari qo‘shimcha ravishda quyidagilar ham kiradi[1]:

– bo‘sh qatorlar soni;
– izohlar soni;

[1] https://habr.com/ru/company/intel/blog/106082/

– izohlar foizi (izohlarni o‘z ichiga olgan satrlar sonining umumiy satrlar soniga nisbati, foizda ifodalangan);

– funksiyalar (sinflar, fayllar) uchun satrlarning o‘rtacha soni;

– funktsiyalar (sinflar, fayllar) uchun manba kodini o‘z ichiga olgan o‘rtacha qatorlar soni;

– modullar uchun o‘rtacha qatorlar soni.

Dastur kodidagi ba’zi ko‘rsatkichlarni hisoblashga asoslangan o‘lchamlar guruhiga Halsted o‘lchamlari kiradi. Ushbu o‘lchamlar quyidagilarga asoslanadi:

η_1 — dasturning takrorlanmas operatorlari soni, jumladan ajratuvchi belgilar, dastur(yoki funktsiya) nomlari va operatsiya belgilari (operatorlar lug‘ati);

η_2 — dasturning takrorlanmas operandlari soni (operandlar lug‘ati);

N_1 — dasturdagi operatorlarning umumiy soni;

N_2 — dasturdagi operandlarning umumiy soni.

Ushbu birlamchi o‘lchov belgilari orqali dastur lug‘atini quyidagicha ifodalash mumkin:

$$\eta = \eta_1 + \eta_2$$

Dasturning uzunligini quyidagicha ifodalash mumkin:

$$N = N_1 + N_2$$

Halsted dastur uzunligini quiyidagi formula bo‘yicha baholashni taklif qildi:

$$N' = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$$

bu yerda N’ - dasturning nazariy uzunligi (stilistik jihatdan to‘g‘ri bo‘lgan dasturlar uchun N ning N’ dan chetlanishi 10% dan oshmaydi.

Dasturning muhim metrik xarakteristikasi uning o‘lchamidir. Dasturning har qanday joriy etilishi uchun mos keladigan metrik o‘lcham xarakteristikasi dasturning hajmi(V- bitlarda ifodalanadi) deb ataladi:

$$V = N * \log_2 \eta$$

NATIJALAR

Halsted o‘lchamlarini tushunish uchun quyidagi misolni ko‘rib chiqamiz:

Misol sifatida pufakchali saralash algoritmining dasturini keltiramiz. Bu dastur C++ dasturlash tilida yozilgan[2].

```
/* Pufakchali saralash algoritmining C++ tilidagi dasturi */
int main()
{ int *arr; int size, temp; cout << "n = "; cin >> size;
```

[2] https://code-live.ru/solutions/cpp/3/

```
  if (size <= 0) { cerr << "Invalid size" << endl;    return 1; }
arr = new int[size];
for (int i = 0; i < size; i++) { cout << "arr[" << i << "] = ";    cin >> arr[i]; }
for (int i = 0; i < size - 1; i++)
    {  for (int j = 0; j < size - i - 1; j++)
          {  if (arr[j] > arr[j + 1])
                { temp = arr[j];     arr[j] = arr[j + 1];  arr[j + 1] = temp;   }  }  }
  for (int i = 0; i < size; i++)  {  cout << arr[i] << " ";  }
  cout << endl;   delete [] arr;
  return 0; }
```

Dasturning takrorlanmas operatorlari (operatorlar lug‘ati) sonini hisoblab chiqamiz. Bu holda ajratuvchi belgilar, dastur(yoki funktsiya) nomlari va operatsiya belgilarini ham qo‘shib hisoblaymiz:

int, main, cout, cin, if, cerr, endl, return, new, for, delete,

1 2 3 4 5 6 7 8 9 10 11

( ) , { } , << , >> , ; , = , [ ] , ++, < , + , - , <=

12 13 14 15 16 17 18 19 20 21 22 23

Demak, dasturning takrorlanmas operatorlari soni:

$\eta_1 = 23$.

Dasturning takrorlanmas operandlari (operandlar lug‘ati) sonini aniqlaymiz:

arr, size, temp, i , j , "n = ", 0, 1, "Invalid size", "arr[", "] = "

1 2 3 4 5 6 7 8 9 10 11

Dasturning takrorlanmas operandlari soni:

$\eta_2 = 11$.

Dastur lug‘ati

$\eta = \eta_1 + \eta_2 = 23 + 11 = 34$ ga teng

Dasturdagi operatorlarning umumiy sonini aniqlaymiz:

$N_1 = 114$

Dasturdagi operandlarning umumiy sonini aniqlaymiz:

$N_2 = 59$

Bu ma’lumotlardan foydalanib, dasturning uzunligini hisoblaymiz:

$N = N_1 + N_2 = 114 + 59 = 173$.

Halsted o‘lchami bo‘yicha dasturning nazariy uzunligini aniqlaymiz:

$N' = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2 = 23 * \log_2 23 + 11 * \log_2 11 =$

$= 23 * 4{,}5236 + 11 * 3{,}4594 = 104{,}0428 + 38{,}0534 = 142{,}0962$

Dastur uzunligini baholash uchun quyidagi muqobil ifodalar ham taqdim etilgan[3]:

$N_J = \log_2(\eta_1!) + \log_2(\eta_2!)$

$N_B = \eta_1 * \log_2\eta_2 + \eta_2 * \log_2\eta_1$

$N_C = \eta_1 * sqrt(\eta_1) + \eta_2 * sqrt(\eta_2)$

$N_S = (\eta * \log_2\eta) / 2$

$$N_J = \log_2(\eta_1!) + \log_2(\eta_2!) = \log_2(23!) + \log_2(11!) =$$
$$= 25{,}2505 + 74{,}4527 = 99{,}7032$$

$$N_B = \eta_1 * \log_2\eta_2 + \eta_2 * \log_2\eta_1 = 23 * \log_2 11 + 11 * \log_2 23 =$$
$$= 23 * 3{,}4594 + 11 * 4{,}5236 = 79{,}5662 + 49{,}7596 = 129{,}3258$$

$$N_C = \eta_1 * \sqrt{\eta_1} + \eta_2 * \sqrt{\eta_2} = 23 * \sqrt{23} + 11 * \sqrt{11} =$$
$$= 23 * \sqrt{23} + 11 * \sqrt{11} = 23 * 4{,}7958 + 11 * 3{,}3166 = 110{,}3034 + 36{,}4826$$
$$= 110{,}3034 + 36{,}4826 = 146{,}786$$

$$N_S = \eta * \frac{\log_2\eta}{2} = 34 * \frac{\log_2 34}{2} = 17 * 2{,}5437 = 43{,}2429$$

Halsted o‘lchamlari asosida dasturning hajmi quyidagiga teng:

$V = N * \log_2\eta = 176 * \log_2 34 = 176 * 5{,}0875 = 871{,}9392$

Potentsial minimal hajm V* masalani kodlash mumkin bo‘lgan eng qisqa dasturning hajmi sifatida aniqlanadi.

$$V^* = (2 + \eta_2) * \log_2(2 + \eta_2) = (2 + 11) * \log_2(2 + 11) = 13 * 3{,}7004 =$$
$$= 13 * 3{,}7004 = 48{,}1057$$

Dasturning murakkablik darajasi yoki xatoga moyilligi dasturdagi takrorlanmas operatorlar soniga proportsionaldir. Dasturning murakkabligi darajasi operandlarning umumiy soni va takrorlanmas operandlar soni o‘rtasidagi nisbatga ham mutanosibdir. Bu shuni anglatadiki, agar dasturda bir xil operandlar ko‘p marta ishlatilsa, u xatolarga ko‘proq moyil bo‘ladi.

$$D = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2} = \frac{23}{2} \times \frac{59}{11} = 11{,}5 \times 5{,}37 = 61{,}68 \approx 62$$

Dastur darajasi dasturning xatoga moyilligining teskarisidir, ya'ni past darajadagi dastur yuqori darajadagi dasturga qaraganda xatolarga ko‘proq moyil bo‘ladi:

[3] https://www.javatpoint.com/software-engineering-halsteads-software-metrics

$$L = \frac{1}{D} = \frac{1}{62} = 0{,}016$$

Dasturchining dasturni ishlab chiqishdagi energiya sarflash darajasi dasturning hajmi va qiyinchilik darajasiga proportsionaldir:

$$E = \frac{V}{L} = D * V = 61{,}68 \times 871{,}9392 = 53781{,}21$$

Kod yozish vaqti dasturchining dasturni ishlab chiqishdagi energiya sarflash darajasiga proportsionaldir. Ushbu miqdorni aniqlash uchun empirik tajribalardan foydalanish mumkin. Halsted energiya sarflash darajasini 18 ga bo‘lish kod yozish vaqtini soniyalarda taxmin qilishini aniqladi.

$$T = \frac{E}{18} = \frac{61{,}68}{18} = 3{,}427\ seconds$$

Taqdim etilgan xatolarning taxminiy soni:

$$B = \frac{E^{\frac{2}{3}}}{3000} = \frac{53781{,}21^{\frac{2}{3}}}{3000} = \frac{1424{,}7994}{3000} = 0{,}4749$$

Halsted tomonidan taqdim etilgan xatolarning soni dasturni ishlab chiqishdagi xatolarning taxminiy soni darjasidir. Dastur faylida xatolar soni darajasi 2 dan kam bo‘lishi kerak. Tajribalar shuni ko‘rsatdiki, C++ dasturlashda manba fayl deyarli har doim taklif qilinganidan ko‘ra ko‘proq xatolarni o‘z ichiga oladi. Nosozliklar soni B ga qaraganda tezroq o‘sadi.

XULOSA

Dasturiy ta’minotning samaradorligini aniqlashda o‘lchovlardan foydalanish ishlab chiqilgan va ishlab chiqilayotgan dasturning murakkabligini o‘rganish, ish hajmini, ishlab chiqilgan dastur uslubini va har bir ishlab chiquvchining ma’lum bir yechimni amalga oshirish uchun sarflagan energiyasini baholash imkonini beradi. Biroq, o‘lchovlar faqat maslahat xarakterida bo‘lib xizmat qilishi mumkin, ularni to‘liq qo‘llab bo‘lmaydi, chunki dasturchilar dasturiy ta’minotni ishlab chiqishda o‘z dasturi uchun u yoki bu o‘lchovni minimallashtirish yoki maksimal darajada oshirishga harakat qilib, dastur samaradorligini pasaytirishgacha bo‘lgan usullarga murojaat qilishlari mumkin. Bundan tashqari, agar, masalan, dasturchi oz sonli kod satrlarini yozgan bo‘lsa yoki oz sonli tarkibiy o‘zgarishlarni amalga oshirgan bo‘lsa, bu uning hech narsa qilmaganligini anglatmaydi, lekin bu dastur nuqsoni juda qiyin bo‘lganligini anglatishi mumkin. Biroq, oxirgi muammoni qisman murakkablik

ko‘rsatkichlari yordamida hal qilish mumkin, chunki murakkabroq dasturda xatoni topish qiyinroq.

REFERENCES

1. Холстед М. Х. Начала науки о программах / Пер. с англ. М.: Финансы и статистика, 1981. 128 с.
2. Кожомбердиева Г.И. Оценка качества программного обеспечения. СПб: Изд-во ПГУПС, 2010. 44 с
3. Черников Б.В., Поклонов Б.Е. Оценка качества программного обеспечения: практикум. М.: ФОРУМ; ИНФРА-М, 2012.400 с.
4. Зубкова Т.М. Технология разработки программного обеспечения. Учебное пособие. — Оренбург: Оренбургский государственный университет, 2017.
5. Гагарина Л.Г. Технология разработки программного обеспечения. / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул. – М.: «ФОРУМ»: ИНФРА-М, 2009.
6. Орлов С.А. Программная инженерия. Учебник для вузов. 5-е издание обновленное и дополненное. – СПб.: Питер, 2016.

Vebsayt

7. www.verifysoft.com/en_halstead_metrics.html
8. https://code-live.ru/solutions/cpp/3/
9. https://habr.com/ru/company/intel/blog/106082/
10. https://www.javatpoint.com/software-engineering-halsteads-software-metrics