# Анализ используемых метрик оценки информационной сложности программ

Программная метрика - это функция, с программными данными в качестве входной величины и числовым значением в качестве выходного [1].

Как правило, метрики сложности делят на три основные группы [6][7]:

* метрики размера ПО;
* метрики сложности потока управления ПО;
* метрики сложности потока данных ПО.

# Метрики размера ПО

## Количество строк кода (LOC)[5]

Выделяют два основных показателя SLOC [6]:

• количество «физических» строк кода (используемые аббревиатуры LOC, SLOC, KLOC, KSLOC, DSLOC) — определяется как общее число строк исходного кода, включая комментарии и пустые строки;

• количество «логических» строк кода SLOC (используемые аббревиатуры LSI, DSI, KDSI, где «SI» — source instructions) — определяется как количество команд и зависит от используемого языка программирования.

Преимущества: Простота

Недостатки:

- вычислительные усилия для строки кода не учитываются в метрике, что затрудняет ее способность давать точную меру сложности программного обеспечения.

- не может быть непосредственно применена к графическим языкам программирования [2]

- зависит от используемого языка программирования [6].

## 2. Меры сложности Холстеда

Меры сложности Холстеда [3] - класс метрик, которые опираются на классификацию программных токенов как операторов или операндов. Дает оценку сложности понимания или написания программы (D) и усилия по написанию программы (E).

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

• NUOprtr (Number of Unique Operators) — число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

• NUOprnd (Number of Unique Operands) — число уникальных операндов программы (словарь операндов);

• Noprtr (Number of Operators) — общее число операторов в программе;

• Noprnd (Number of Operands) — общее число операндов в программе.

На основании этих характеристик рассчитываются оценки:

• словарь программы (Halstead Program Vocabulary, HPVoc):

HPVoc = NUOprtr + NUOprnd;

• длина программы (Halstead Program Length, HPLen):

HPLen = Noprtr + Noprnd;

• объем программы (Halstead Program Volume, HPVol):

HPVol = HPLen log2 HPVoc;

• сложность программы (Halstead Difficulty, HDiff):

HDiff = (NUOprtr/2) x (NOprnd / NUOprnd);

Также на основе показателя HDiff имеется возможность оценивать усилия программиста при разработке при помощи показателя HEff (Halstead Effort): HEff= HDiff x HPVol.

# Метрики сложности потока управления ПО

Метрики второй группы базируются на анализе управляющего графа программы. Управляющий граф G(V, Е) строится в виде ориентированного графа, в котором вычислительные операторы или выражения представляются в виде вершин V(Vh, ..., Vm), а передача управления между вершинами — в виде ребер Е(ЕХ,Еп) [6].

Также к метрикам потока управления относятся, например, метрика Майерса (расширение метрики Маккейба), метрика Джилба, основанная на определении сложности программы как ее насыщенность выражениями типа IF-THEN-ELSE, метрика граничных значений и некоторые другие.

## Метрика Маккейба

Одной из часто используемых метрик данной группы является метрика Маккейба. Показатель цикломатической сложности вычисляется на основе графа управляющей логики программы (control flow graph)[6].

Цикломатическая сложность Маккейба (CC) [4] - указывает количество линейно независимых путей в потоке управления. Подсчитывает количество решений в программе или, соответственно, количество ребер и узлов в графическом представлении.

Упрощенная формула вычисления цикломатической сложности имеет следующий вид:

С = е - п + 2, где е — число ребер; п — число узлов на графе управляющей логики.

Формула:

https://studme.org/htm/img/15/1573/21.png

где т — количество дуг (ребер) ориентированного графа G; п — количество вершин; г —количество компонент связности графа[8].

Достоинства[7]:

- простота ее вычисления;

- повторяемость результата;

- наглядность и содержательность интерпретации.

Недостатки [7]:

- нечувствительность к размеру ПО;

- нечувствительность к изменению структуры ПО;

- отсутствие корреляции со структурированностью ПО;

- отсутствие различия между конструкциями «развилка» и «цикл», отсутствие чувствительности к вложенности циклов.

## Метрика Джилба

Одной из наиболее простых, но достаточно эффективных оценок сложности программ является метрика Т. Джилба, в которой логическая сложность программы определяется как насыщенность программы выражениями типа IF – THEN – ELSE. При этом вводятся две характеристики CL – абсолютная сложность программы, характеризующаяся количеством операторов условия[12];

CL = Nу,

где Nу- число операторов условия типа IF – THEN – ELSE

и cl – относительная сложность программы, характеризующаяся количеством операторов условия, т.е. cl определяется как отношение cl к общему числу операторов.

Ci = Cl/N, где N – общее количество операторов.

## Метрика «граничных значений» оценки сложности

Введем несколько дополнительных понятий, связанных с графом программы.

G = (V, E) – ориентированный граф программы с единственной начальной и единственной конечной вершинами. В этом графе число входящих в вершину дуг называется отрицательной степенью вершины, а число исходящих из вершины дуг называется положительной степенью вершины. Тогда набор вершин графа можно разбить на 2 группы [12]:

1. Принимающие вершины – те, у которых положительная степень ≤ 1;

2. Вершины отбора – вершины, , у которой положительная степень ≥ 2.

Для получения оценки необходимо разбить исходный граф G на макс. число подграфов G’, удовлетворяющим следующим условиям:

- Вход в подграф осуществляется через вершину отбора.

- Каждый подграф включает вершину, называемую в дальнейшем «нижняя граница подграфа», в которую можно попасть из любой другой вершины подграфа.

Число вершин, образующих такой подграф, равно скорректированной сложности вершины отбора. Каждая принимающая вершина имеет скорректированную сложность равную 1, кроме конечной вершины, скорректированная сложность которой равна 0.

Скорректированные сложности всех вершин графа G суммируются, образуя абсолютную граничную сложность. После этого определяется относительная граничная сложность программы:

https://konspekta.net/infopediasu/baza14/274751895339.files/image444.gif

где S0 – относительная граничная сложность программы; Sа – абсолютная граничная сложность программы; v – общее число вершин графа программы.

# Метрики сложности потока данных ПО

Метрики сложности потока данных базируются на оценке использования, конфигурации и размещения данных в программе. В первую очередь это касается глобальных переменных. К данной группе относятся метрики Чепина, Спена, метрика обращения к глобальным переменным [6].

## Метрика Чепина

Существует несколько модификаций Метрики Чепина. Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода [6][10].

Все множество переменных, составляющих список ввода-вывода, разбивается на четыре функциональные группы[6].

1. Множество «Р» — вводимые переменные для расчетов и для обеспечения вывода. Примером может служить используемая в программах лексического анализатора переменная, содержащая строку исходного текста программы, т. е. сама переменная не модифицируется, а только содержит исходную информацию.

2. Множество «М» — модифицируемые или создаваемые внутри программы переменные.

3. Множество «С» — переменные, участвующие в управлении работой программного модуля (управляющие переменные).

4. Множество «Т» — неиспользуемые в программе («паразитные») переменные. Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе.

Далее вводится значение метрики Чепина:

https://studref.com/htm/img/15/6619/473.png

При этом а1, а2, а3, а4 – весовые коэффициенты. С учетом их значений получаем:

https://studref.com/htm/img/15/6619/476.png

## Метрика спена.

Определение спена основывается на локализации обращений к данным внутри каждой программной секции. Спен — это количество утверждений, содержащих данный идентификатор, между его первым и последним появлением в тексте программы. Следовательно, идентификатор, появившийся п раз, имеет спен, равный п — 1. При большом спене усложняется тестирование и отладка [8].

## Метрика Кафура

Метрика Кафура — метрика, основанная на учете потока данных [11]. Вводятся понятия локального и глобального потока.

Локальный поток информации из Л в В существует, если:

* модуль А вызывает модуль В (прямой локальный поток);
* модуль В вызывает модуль А и модуль А возвращает модулю В значение, которое используется в В (непрямой локальный поток);
* модуль С вызывает модули А, В и передает результат выполнения модуля А в модуль В.

Глобальный поток информации из модуля А в модуль В через глобальную структуру данных D существует, если модуль А помещает информацию в D, а модуль В использует информацию из D. На основе этих понятий вводится величина I — информационная сложность процедуры. length — сложность текста процедуры (измеряется через какую- нибудь из метрик объема, типа метрик Холстеда, МакКейба, LOC и т.п.); fan\_in — количество локальных потоков внутрь процедуры плюс количество структур данных, из которых процедура берет информацию; fan\_out — количество локальных потоков из процедуры плюс количество структур данных, которые обновляются этой процедурой.

Можно определить информационную сложность модуля как сумму информационных сложностей входящих в него процедур.

Следующий шаг — определение информационной сложности модуля относительно некоторой структуры данных. Информационная мера сложности модуля относительно структуры данных представима как

https://studme.org/htm/img/15/1573/24.png

где W — число процедур, которые только обновляют структуру данных; R — число процедур, которые только читают информацию из структуры данных; WrRd — число процедур, которые и читают, и обновляют информацию в структуре данных.

# Список литературы

1. T. Honglei, S. Wei, and Z. Yanan, “The research on soft-ware metrics and software complexity metrics”, in 2009 International Forum on Computer Science-Technology and Applications, vol. 1, Dec. 2009, pp. 131–136.
2. L. Capitán and B. Vogel-Heuser, “Metrics for software quality in automated production systems as an indicator for technical debt”, in 2017 13th IEEE Conference on Automation Science and Engineering (CASE), Aug. 2017, pp. 709–716.
3. M. H. Halstead, “Elements of software science”,Else-vier New York, 1977.
4. T. J. McCabe, “A complexity measure”, in Proceedings of the 2Nd International Conference on Software Engi-neering, ser. ICSE ’76, San Francisco, California, USA: IEEE Computer Society Press, 1976, pp. 407–.
5. <https://publik.tuwien.ac.at/files/publik_271875.pdf>
6. <https://studref.com/388370/informatika/metriki_slozhnosti_programmnogo_obespecheniya_informatsionnyh_sistem> (Основы теории надежности информационных систем)
7. <https://studme.org/128174/informatika/metriki_slozhnosti_potoka_upravleniya_programm> (Надежность и безопасность программного обеспечния)
8. Милютин, А. Метрики кода программного обеспечения [Электронный ресурс] / А. Милютин. — URL: <http://www.viva64.eom/ru/a/0045/>
9. Watson, Arthur Н. Structured Testing: ATesting Methodology Using Cyclomatic Complexity Metric [Электронный ресурс] / Arthur H. Watson, Thomas J. McCabe //
10. Новичков, А. Метрики кода и их практическая реализация в IBM Rational [Электронный ресурс] / А. Новичков. — URL: http://www.viva64.com/ go.php?url=241.
11. Kafura, D. The Use of Software Complexity Metrics in Software Maintenance / D. Kafura, G. Reddy // IEEE Transactions on Software Engineering. —1987. — March.
12. <https://infopedia.su/14x16a65.html>