«Информатика, кибернетика и вычислительная техника»

Научная статья
УДК 004.412
DOI: 10.17072/1993-0550-2023-3-76-84

Разработка приложения для получения метрик программного продукта на языке объектно-ориентированного программирования

Артем Олегович Корзников[1], Наталья Николаевна Дацун[2]

[1, 2]Пермский государственный национальный исследовательский университет, Пермь, Россия
[1]artemkorz@mail.ru, https://orcid.org/0009-0006-3941-9214
[2]nndatsun@inbox.ru, https://orcid.org/0000-0001-8560-7036

Аннотация. Все группы процессов жизненного цикла программного продукта на стороне разработчика сложны в осуществлении. При этом следует учитывать возможность генерации программного кода, а в случае командной работы – потребность оценки вклада каждого ее участника. В работе предлагается количественная оценка различных аспектов программного обеспечения путем вычисления метрик программного кода. Цель данной работы – разработка приложения расчета метрик для различных языков объектно-ориентированного программирования (ООП). Задачами являются разработка подходов к применению метрик для оценки и сравнения программного кода, реализация приложения расчета метрик. Создано описание для подмножества языков C#, C++ и Java. Впервые предложены шкалы значений для метрик Холстеда, подходы к анализу динамики изменения программного продукта и сравнению различных программ решения одной задачи. Это позволяет дать интерпретацию значений метрик. Разработано приложение Metrics Observer расчета 11 метрик для программ на языке ООП и 16 метрик, не зависящих от парадигмы. Практическая значимость состоит в подготовке решений для сравнения различных реализаций одной задачи, выявления участков кода для рефакторинга, оценки динамики изменения качества кода в процессе разработки / рефакторинга и вклада в проект отдельных разработчиков.

Ключевые слова: *метрика; язык объектно-ориентированного программирования; статический анализ кода; оценка программного продукта*

Для цитирования: *Корзников А.О., Дацун Н.Н.* Разработка приложения для получения метрик программного продукта на языке объектно-ориентированного программирования // Вестник Пермского университета. Математика. Механика. Информатика. 2023. Вып. 3(62). С. 76–84. DOI: 10.17072/1993-0550-2023-3-76-84.

Статья поступила в редакцию 05.07.2023; одобрена после рецензирования 31.07.2023; принята к публикации 15.09.2023.

«Computer Science, Cybernetics and Computing»

Research article

Program Realization for Code Metrics Calculation in Object-Oriented Programming Language

Artem O. Korznikov[1], Natalya N. Datsun[2]

[1,2]Perm State University, Perm, Russia
[1]artemkorz@mail.ru, https://orcid.org/0009-0006-3941-9214
[2]nndatsun@inbox.ru, https://orcid.org/0000-0001-8560-7036

Abstract. All groups of software product lifecycle processes executed by developers are complicated to implement. The code generation possibility and a requirement of evaluation for each participant contribution in case of teamwork also should be considered. A quantitative estimation for various aspects of software is proposed by code metrics calculation. The work purpose is development of a program for calculating the metrics of a software product for various object-oriented programming languages. The tasks are development of metrics exploitation approaches for evaluating and comparing code and implementation of the metric calculation program. The description for subsets of the languages C#, C++ and Java was created. A values gradation Halstead metrics and the approaches for analyzing dynamic changes of a software product and comparison of different programs solving the same problem are proposed and first allowed calculated metrics values interpretation. "Metrics Observer" program was developed. It calculates 11 metrics values for programs in object-oriented language and 16 metrics values that do not depend on the paradigm. Practical significance is to prepare solutions for comparing different implementations of the same task, identifying code modules for refactoring, estimating dynamic code quality changes during the development / refactoring process and individual contributions of developers to a project.

Keywords: *metrics; object-oriented programming language; static code analysis; software product evaluation*

For citation: *Korznikov A.O., Datsun N.N.* Program Realization for Code Metrics Calculation in Object-Oriented Programming Language. Bulletin of Perm University. Mathematics. Mechanics. Computer Science. 2023;3(62):76-84. (In Russ.). DOI: 10.17072/1993-0550-2023-3-76-84.

The article was submitted 05.07.2023; approved after reviewing 31.07.2023; accepted for publication 15.09.2023.

Введение

При постоянном совершенствования и обновлении информационных технологий (ИТ) – цифровая экономика, решения на основе искусственного интеллекта, всепроникающие ИТ и т. д. – зачастую затруднительно корректно определить экономические характеристики проекта. В настоящее время метрики широко применяются в программной инженерии [1] при оценивании труда программистов: объема, качества и сложности проделанной работы на различных этапах жизненного цикла программного продукта. Кроме того, отраслевые стандарты, такие как ISO 9000 [2] и модели, например, CMM [3] и CMMI [4], также включают работу с метриками.

Неоспоримым достоинством метрик является их четкая определенность и универсальность. Метрика программного продукта – это формальная мера, выраженная числовым значением, определяющая некоторое свойство программы [5]. Они позволяют лицам, принимающим решения, оценивать трудовые, финансовые и временные затраты для решаемой задачи, а программистам – выявлять проблемные участки разрабатываемой системы, сравнивать различные варианты решения и улучшать код.

Однако некоторыми недостатками обладает результат вычисления метрики, являющийся количественной оценкой определенного показателя. На полученное значение могут оказать влияние, например, стиль написания программы, в том числе использование определенного шаблона проектирования. Помимо этого, программный код может быть намеренно изменен для искусственного завышения результатов вычислений, что говорит об отсутствии объективности. Поэтому необходимым становится рассмотрение метрик в совокупности и в динамике, а интерпретация полученного результата также является нетривиальной задачей.

В табл. 1 приведено сравнение систем для расчета метрик с разработанным приложением Metrics Observer.

Программ, выполняющих аналогичные и похожие задачи, небольшое количество. Большинство из программ, вычисляющих значения метрик, работают с фиксированным набором из нескольких языков. Программы [6–8] вычисляют ограниченный набор метрик.

В данной работе реализовано приложение Metrics Observer, не зависящее от описания конкретного языка. В используемый набор метрик [9] входят стандартные метрики (строки кода, оценка стилистики), метрики Холстеда, а также метрики кода объектно-ориентированных программ: метрики Чидамбера и Кемерера, метрики Лоренца и Кидда. Разработано описание языка объектно-ориентированного программирования (ООП) при помощи двух файлов. Составлено описание для подмножества языков C#, C++ и Java. Приведены примеры использования приложения.

1. Анализ языков ООП

Различные языки программирования отличаются лексикой, синтаксисом и семантикой.

Набор лексем можно условно разделить на операнды и операторы: ключевые слова и специальные символы.

Для различных языков их состав не идентичен, причем одни и те же элементы лексики в некоторых из них могут являться, например, операторами, в иных – операндами, а также это может зависеть от контекста в рамках одного языка. Исходя из этого, необходимо четкое задание списка зарезервированных слов языка еще на этапе лексического анализа для его корректного осуществления. В этом разделе обсуждаются объектно-ориентированные языки C#, Java, C++ и Python. На рис. 1 показано сравнение состава лексем для них.

Таблица 1. *Сравнение систем для расчета метрик программных продуктов*

Название системы, источник	Язык анализируемых программ	Количество метрик, расчет которых реализован в системе			
		Стандартные метрики	Метрики ООП	Метрики потока управления	Метрики стилистики
MS Visual Studio [6]	C#	2	2	1	1
Source Stat [7]	C, C++	2	0	0	1
NLOC [8]	C, C++, C#, Java, Visual Basic, Pascal, Delphi	2	0	0	1
Metrics [10]	C#	16	11	0	1
Metrics Observer	C#, C++, Java	16	11	0	1

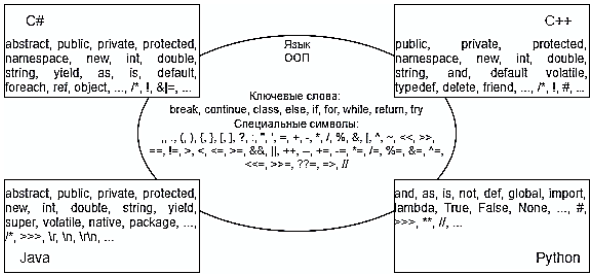

Рис. 1. *Обобщение и специализация ключевых слов и зарезервированных символов языков ООП*

Ввиду принадлежности рассматриваемых языков к одной парадигме программирования, состав синтаксических конструкций языка является схожим. Однако каждый язык имеет собственные особенности их описания.

В табл. 2 приведено описание некоторых из структурных единиц языков ООП. Так, в C# возможно отдельное описание свойств в качестве элементов класса, в то время как в остальных языках допускается лишь создание соответствующих методов. В то же время, по сравнению с C#, языки обладают более сложным синтаксисом. Например, C++ допускает множественное наследование, обладает более сложным описанием связи дочернего и родительского класса.

Таблица 2. *Сравнение синтаксических конструкций объектно-ориентированных языков*

Язык ООП	Реализация наследования
C#	<заголовок класса> ::= class <имя> [":" <имя базового класса>]
C++	<заголовок класса> ::= class <имя> [":" [<модификаторы>] <имя базового класса> ("," <имя базового класса>)]
Python	<заголовок класса> ::= class <имя> ["(" <имя базового класса> ("," <имя базового класса>) ")"]

Помимо этого, в одной и той же синтаксической структуре могут присутствовать разные лексемы, которые имеют идентичное предназначение (например, символ ":" и ключевое слово "extends" в различных языках обозначают, что далее будет записан базовый класс при описании наследования). Кроме того, некоторые из лексем могут быть парными, такие как открывающие и закрывающие скобки.

Существуют и другие отличия семантики объектно-ориентированных языков, оказывающие влияние на интерпретацию кода. В том числе, для корректного определения теку-

щего элемента программы необходимо учитывать семантику конструкторов, имеющих имя, совпадающее с именем класса. Необходимо корректно распознавать и общие для всех языков программирования элементы, например, выражения, операторы, идентификаторы. Также для вычисления значений метрик ООП необходимо определять такие общие элементы, как пространства имен, классы и блоки кода, находить и запоминать некоторые их характеристики. Помимо этого, семантика оказывает влияние и на расчет промежуточных значений метрик, например, полное имя записывается с помощью названий пространств имен и классов (объектов).

2. Обобщенное определение синтаксиса языков ООП

Для извлечения определенных конструкций языка, необходимых при вычислении значений набора метрик, составлено формальное описание. Исходя из поставленной задачи, проверяемая программа предполагается компилируемой, задачей структур описания языка является их выделение из кода для последующего анализа. Синтаксис определен при помощи форм Бэкуса–Наура для универсального задания для трех объектно-ориентированных языков: C#, Java и C++. Фрагмент набора БНФ представлен на рис. 2.

<программа> ::= {<оператор добавления ресурса>}<элемент программы> {<элемент программы>}
<оператор добавления ресурса> ::= <initExternalSyKw> <оператор добавления ресурса CS> | <оператор добавления ресурса JAVA> | <оператор добавления ресурса CPP>
<оператор добавления ресурса CS> ::= <выражение>
<оператор добавления ресурса JAVA> ::= <модификаторы> <выражение>
<оператор добавления ресурса CPP> ::= "<" <имя> ">" | <константа> | namespace <имя> ";"
<элемент программы> ::= <пространство имен> | <класс> | <перечисление> | <метод>

Рис. 2. *Пример БНФ*

3. Описание лексики и связанной семантики языка

Описание лексики языка формируется при помощи json-файла по определенным правилам. Было построено описание для подмножеств объектно-ориентированных языков программирования: C# 7.3, Java SE 8, C++ 11.

Описание лексики языка включает определение ключевых слов и специальных символов языка. К ключевым словам ("Keywords") отнесены лексемы, являющиеся зарезервированными. К лексемам специальных символов ("Symbols") отнесены любые лексемы, не содержащие буквы алфавита кириллицы и латиницы, арабские цифры.

К семантике, связанной с лексикой, относится разделение лексем на группы ("Groups") в соответствии с выполняемой задачей в коде и задание пар лексем ("Pairs"). Для определения группы необходимо указать название ("Name") и список значений, вошедших в ее состав ("Values"). Пары лексем определяются при помощи двух значений лексем ("Name" и "Linked"). Также файл лексики хранит дополнительную информацию о языке: список расширений файлов кода программ, ("Extensions"), список расширений файлов, которые необходимо исключить при поиске файлов языка ("Except") и имя файла синтаксиса языка ("Syntax"). Общий вид файла можно увидеть на рис. 3.

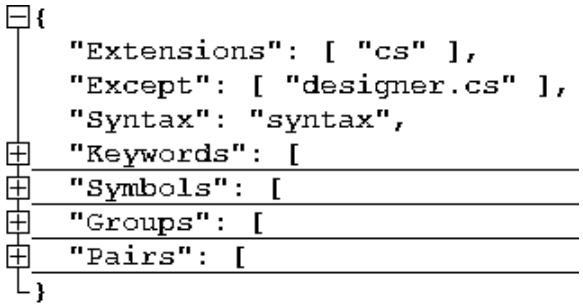

Рис. 3. *Общий вид файла описания лексики*

4. Структуры данных описания кода

Для расчета отобранных метрик необходимо определить значения, участвующие в вычислениях. Они представлены в четырех классах, описывающих код анализируемых программ.

Класс "Блок кода" – базовая структурная единица описания кода программы на объектно-ориентированном языке, которая содержит: название структуры, список полей, список методов, словарь лексем. Кроме того, класс обладает методом для слияния словарей лексем.

Класс "Элемент описания кода", являющийся наследником класса "Блок кода", хранит списки полей и методов. Помимо этого, класс обладает дополнительными атрибутами, которые представляют: список блоков кода, текущий уровень в иерархии классов, список дочерних классов, количество переопределяемых

подклассом методов, количество конструкторов. Кроме того, класс включает список блоков кода и классов, являющихся непосредственными потомками, что позволяет произвести расчет метрик для программ языков ООП.

"Корневой класс" отличается от предыдущего элемента описания отсутствием родительского класса при моделировании иерархии наследования. Данный класс сохраняет характеристики класса "Элемент описания кода", дополняя их глубиной дерева наследования, отвечает за получение значения данной метрики.

"Пространство имен" – элемент описания кода, который может хранить классы. Наследует от класса поля, хранящие элементы структур. Содержит списки: подпространств имен и классов. Данный класс выполняет преобразование списка классов в их древовидные структуры, соответствующие иерархии наследования, сохраняя на верхнем уровне корневые классы.

В результате описание анализируемой программы образует дерево. Корнем является пространство имен проекта, содержащее пространства имен и их подпространства. Узлы этого уровня включают в себя корневые классы, содержащие дочерние классы. Листья являются блоками кода.

5. Описание синтаксиса языка и связанной семантики

Файл xml-формата использован для определения синтаксиса и необходимой для расчета отобранного набора метрик семантики языка.

Для задания правил построения описания составлены БНФ, пример которых представлен на рис. 4. Также этот файл задает логику для расчета промежуточных значений метрик.

<описание> ::= <xmlver> <ArrayOfElement>
<ArrayOfElement> ::= <arrayTag> <Program> {<Element>} <arrayTagClose>
<Program> ::= <ElementTag> <NameTag> "program" <NameTagClose> <Descriptrion> <ElementTagClose>
<Element> ::= <ElementTag> <NameValue> <Variables> <Descriptrion> <Return> <ElementTagClose>
<Descriptrion> ::= <DescriptrionTag> {<Entity>} <DescriptrionTagClose>
<Entity> ::= <EntityTag> (<Condition> | <Instruction> | <Reserved> | <ProgramElement>) <EntityTagClose>

Рис. 4. *БНФ описания файла описания синтаксиса*

6. Алгоритм работы приложения

Алгоритм работы приложения (рис. 5) включает нахождение файлов проекта, подлежащих анализу, описания соответствующих языков программирования, статический анализ кода [11] анализируемого приложения и расчет метрик по результатам анализа.

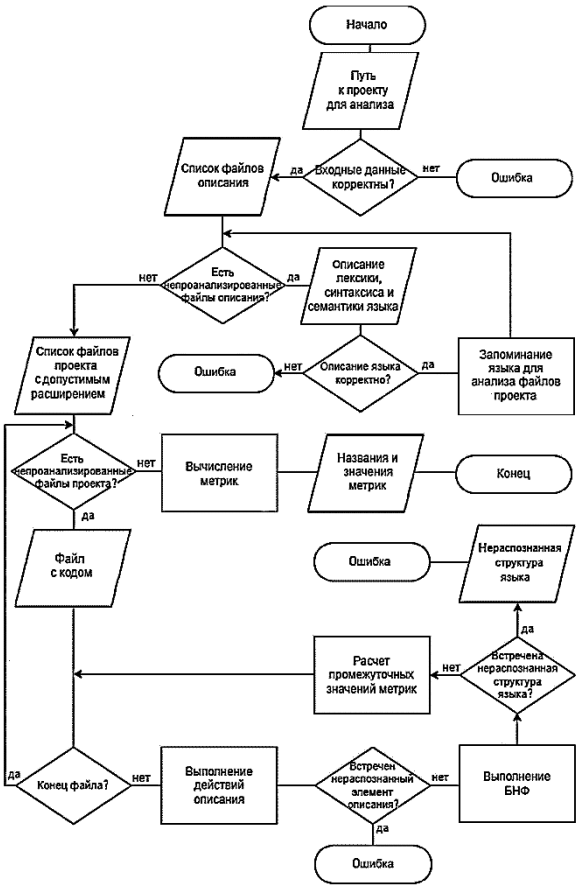

Рис. 5. *Алгоритм работы приложения*

7. Анализ программного продукта по значениям метрик

На рис. 6 представлены вычисленные метрики проекта Metrics Observer.

Комментарий к полученным результатам приведен на рис. 7. Он составлен для метрик, значения которых можно сравнить с определенной теоретической нормой.

Значение метрики "Оценка количества комментариев" является отрицательным, поэтому программу можно считать не задокументированной.

Metrics Observer.xlsx

	A	B	C
1	**Проект "Metrics Observer"**		
2	Стандартные метрики	Физические строки кода	4498
3		Логические строки кода	12638
4		Оценка стилистики программы	-9
5	Метрики Холстеда	Длина программы	18090
6		Словарь программы	1527
7		Объем программы	191328.6
8		Уровень качества программирования	0.000592
9		Трудоемкость кодирования программы	1688.21
10		Теоретическая длина программы	14684.75
11		Теоретический словарь программы	306
12		Теоретический объем программы	121257.66
13		Оценка уровня качества программирования	0.634
14		Сложность	297.78
15		Нижняя работа по программирования	301891.31
16		Верхняя работа по программированию	323003258.6
17		Нижнее время программирования	4.66
18		Верхнее время программирования	4984.62
19		Нижняя оценка количества ошибок	1.5
20		Верхняя оценка количества ошибок	156.92
21	Метрики Чидамбера и Кемерера	Взвешенные методы на класс	6.03
22		Высота дерева наследования	6
23		Количество дочерних классов	17
24		Сцепление между классами	2.45
25		Отклик класса	13.94
26		Число аргументов метода	0.36
27		Недостаток связности в методах	32.39
28	Метрики Лоренца и Кидда	Размер класса	7.98
29		Переопределяемые подклассом операции	0.102
30		Высота дерева наследования	6
31		Добавляемые подклассом операции	2.67
32		Индекс специализации	0.0667

Рис. 6. *Вычисленные значения метрик для проекта Metrics Observer*

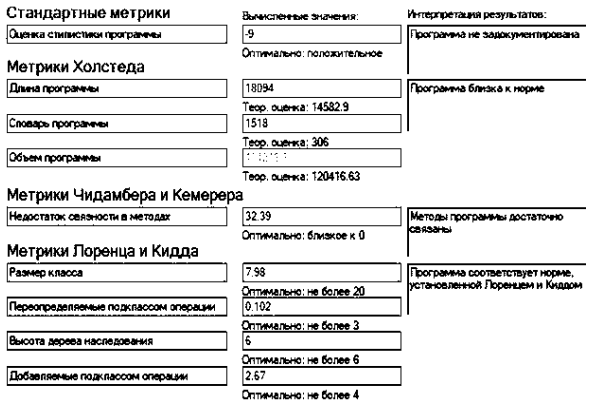

Рис. 7. *Комментарий к вычисленным значениям*

Значение метрики "Длина программы" меньше теоретической оценки, "Словарь программы" превосходит теоретическую оценку и "Объем программы" близок к теоретической оценке. Исходя из полученных результатов, программа не соответствует норме полностью, однако ее можно считать близкой к норме.

Существует недостаток связности в методах, но он лежит в пределах допустимого, поэтому методы программы достаточно связаны. Все рассматриваемые метрики Лоренца и Кидда соответствуют оптимальным значениям, поэтому программа соответствует норме, установленной для данной группы метрик.

8. Анализ динамики изменения программного продукта

Для анализа использовались различные версии одного приложения. Для каждой версии были определены значения метрик и максимальные значения из полученных для всех рассматриваемых версий, которые указаны в табл. 3. Определена доля каждого значения метрики от максимального, и проведено сравнение с помощью диаграммы (рис. 8).

Количество логических строк кода отражает размер программы. Рост показателей метрик "Оценка уровня качества программирования" и "Индекс специализации" является желательным, "Сложность" и "Верхняя оценка количества ошибок" – нежелательным.

Таблица 3. *Вычисленные метрики для различных версий программного продукта*

Метрика	Версия программного продукта			
	№1	№2	№3	№4
Логические строки кода	12706	13492	13162	13633
Оценка уровня качества программирования	0.553	0.555	0.548	0.559
Сложность	245.9	249.8	241.2	243.8
Верхняя оценка количества ошибок	138.5	145.6	144.6	140.7
Сцепление между классами	8.84	7.57	7.43	7.43
Отклик класса	35.3	33.8	34	33.5
Недостаток связности в методах	191.3	117.8	87.6	81.5
Индекс специализации	0.179	0.425	0.425	0.425

Для значений других метрик необходимо некоторое оптимальное значение.

Метрики "Сцепление между классами", "Отклик класса", "Недостаток связности в методах" и "Индекс специализации" относятся к метрикам ООП.

Метрика "Индекс специализации" связана с иерархией наследования.

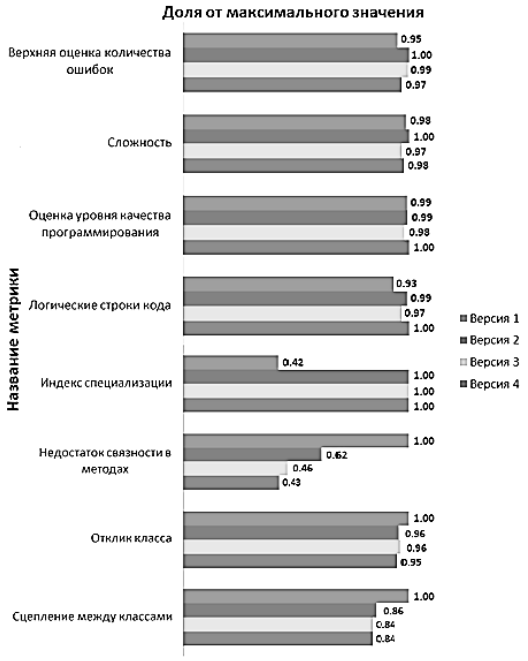

Рис. 8. *Изменение значений метрик в зависимости от версии приложения*

Анализ полученных значений метрик показал, что на протяжении изменений в версиях 1–4 снижаются значения метрик "Сцепление между классами" и "Недостаток связности в методах".

В результате внесения изменений между версиями 1 и 2 повысилось значение метрик "Индекс специализации", "Верхняя оценка количества ошибок" и "Сложность",

После внесения изменений возросли значения метрик "Логические строки кода", "Оценка уровня качества программирования", "Верхняя оценка количества ошибок" по отношению к изначальным и понизилось значение "Сложность".

Поэтому можно сделать следующие выводы. Изменения потребовали расширения программы, однако, наблюдается положительная тенденция изменений. Структура приложения грамотно спроектирована. Изменения, внесенные между версиями 1 и 2, были направлены на разделение логики родительских и дочерних классов и могли содержать ошибки.

9. Анализ программного продукта на основании средних значений метрик

Для анализа выбраны различные программы, решающие одну задачу. Для каждой из программ определены значения метрик, для программ А–В определено среднее арифметическое (табл. 4). Определена доля каждого значения метрики от среднего значения для сравнения с помощью диаграммы (рис. 9).

У анализируемой программы Г показатели метрик "Объем программы", "Сложность", "Верхнее время программирования", рост которых не желателен, ниже средних для аналогичных программ. При этом высота дерева наследования не отличается от средней, однако, ниже число добавляемых подклассом операций, индекс специализации существенно отличается.

Таблица 4. *Вычисленные метрики для различных версий программного продукта*

Метрика	Программный продукт			Среднее	Прог. Г
	А	Б	В		
Объем программы	730	644	719	698	322
Верхнее время программирования	0.27	0.25	0.28	0.27	0.073
Сложность	14.9	15.8	15.4	15.36	11.9
Высота дерева наследования	2	2	2	2	2
Добавляемые подклассом операции	1	2	1	1.33	1
Индекс специализации	0.5	0.33	0.5	0.44	0

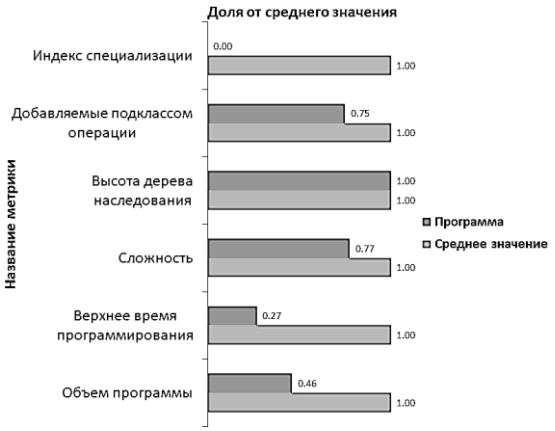

Рис. 9. *Сравнение со средним значением для программы Г*

На основании наблюдений можно сделать вывод: приложение Г является более простым для понимания и реализации, в то же время, классы в иерархии наследования сильно

связаны друг с другом и меньшее число методов определяются в дочерних классах, что может говорить о недостатках проектирования.

Заключение

Проанализированы современные языки объектно-ориентированного программирования и составлен набор метрик для комплексного анализа свойств программного продукта на этих языках. Выполнен анализ доступных приложений для расчета метрик.

Определен обобщенный синтаксис подмножеств объектно-ориентированных языков C#, C++ и Java с помощью БНФ. Разработано описание языка объектно-ориентированного программирования при помощи файлов описания лексики и синтаксиса и связанной с ними семантики языка, необходимой для расчета метрик. Структура файла описания синтаксиса задана при помощи БНФ. Приложение Metrics Observer спроектировано независимо от описания конкретного языка из рассматриваемых.

Реализация выполнена на языке C#. Создано 67 классов. Описание поддерживаемых языков включает 4 файла: 3 json-файла описания лексики и 1 xml-файл описания синтаксиса. Разработка приложения выполнена в среде Microsoft Visual Studio 2019. Размер файла приложения: 121 КБ.

Приложение Metrics Observer позволяет вычислить 11 метрик ООП и 16 метрик, не зависящих от парадигмы. Тестирование проведено для программного кода проекта Metrics Observer (язык C#) и примеров программ на языках C++ и Java на трех испытательных стендах.

В дальнейшем возможно провести тестирование для проектов из различных предметных областей для дополнения списка элементов языков, используемых в исходном коде, а также создать инструмент автоматизации формирования описания языка.

Кроме того, возможно создание приложения для интерпретации вычисленных значений метрик в зависимости от потребности пользователя.

Список источников

1. *Xenos M.* Software Metrics and Measurements // Encyclopedia of E-Commerce, E-Government and Mobile Commerce. Idea Group Publishing, 2006. P. 1029–1036.
2. *ISO* 9000 family. Quality management. URL: https://www.iso.org/iso-9001-quality-management.html (дата обращения: 30.06.2023).
3. *CMM.* Capability Maturity Model. URL: https://www.geeksforgeeks.org/software-engineering-capability-maturity-model-cmm/ (дата обращения: 30.06.2023)
4. *CMMI.* Capability Maturity Model Integra-tion. URL: https://docs.microsoft.com/en-us/azure/-devops/boards/work-items/guidance/cmmi/-guidance-background-to-cmmi?view=azure-devops (дата обращения: 30.06.2023).
5. *Звездин С.* Метрики как средство управления качеством // Открытые системы. СУБД. 2009. № 08. С. 36–40.
6. *Значения* метрик кода – Visual Studio (Windows). URL: https://docs.microsoft. com/ru-ru/visualstudio/code-quality/code-metrics-values?view=vs-2019 (дата обращения: 30.06.2023).
7. *SourceStat* – расчет метрик программного обеспечения. URL: http://bitaks.com/products/sourcestat/sourcestat.html (дата обращения: 30.06.2023).
8. *NLOC* – Source Line Counter Tool. URL: http://nloc.sourceforge.net/index.html (дата обращения: 30.06.2023).
9. *Lee M.-C., Chang T.* Software Measurement and Software Metrics in Software Quality // International Journal of Software Engineering and Its Applications. 2013. Vol. 7, № 4. P. 15–34. URL: https://www.researchgate.net/publication/260480820_Software_measurement_and_software_metrics_in_software_quality (дата обращения: 30.06.2023).
10. *Корзников А.О., Дацун Н.Н.* Реализация приложения расчета метрик кода на объектно-ориентированном языке программирования / Актуальные проблемы математики, механики и информатики: сб. статей по материалам студ. конф. / Перм. гос. нац. исслед. ун-т. Пермь, 2022. С. 40–45. URL: https://www.elibrary.ru/item.asp?id=49889579 (дата обращения: 30.06.2023).
11. *Ахо А., Сети Р., Ульман Д.* Компиляторы: Принципы, технологии, инструменты. М.: Вильямс, 2008. 1184 с.

References

1. *Xenos M.* Software Metrics and Measurements. Encyclopedia of E-Commerce, E-Government and Mobile Commerce. Idea Group Publishing, 2006:1029–1036.

2. *ISO* 9000 family. Quality management. URL: https://www.iso.org/iso-9001-quality-management.htm.
3. *CMM*. Capability Maturity Model. URL: https://www.geeksforgeeks.org/software-engineering-capability-maturity-model-cmm/.
4. *CMMI*. Capability Maturity Model Integration. URL: https://docs.microsoft.com/en-us/azure/devops/boards/work-items/guidance/cmmi/guidance-background-to-cmmi?view=azure-devops.
5. *Zvezdin S.* Metrics as means of quality management. Open systems. DBMS. 2009;(08).36-40. (In Russ.).
6. *Znacheniya* metrik koda – Visual Studio (Windows). URL: https://docs.microsoft.com/ru-ru/visualstudio/code-quality/code-metrics-values?view=vs-2019. (In Russ.).
7. *SourceStat* – raschet metrik programmnogo obespecheniya. URL: http://bitaks.com/products/sourcestat/sourcestat.html. (In Russ.).
8. *NLOC* – Source Line Counter Tool. URL: http://nloc.sourceforge.net/index.html.
9. *Lee M.-C., Chang T.* Software Measurement and Software Metrics in Software Quality. International Journal of Software Engineering and Its Applications. 2013;7(4):15-34. URL: https://www.researchgate.net/publication/260480820_Software_measurement_and_software_metrics_in_software_quality.
10. *Korznikov A.O., Datsun N.N.* Realizaciya prilozheniya rascheta metrik koda na ob''ektno-orientirovannom yazyke programmirovaniya / Aktual'nye problemy matematiki, mekhaniki i informatiki: sb. statej po materialam stud. konf. / Perm. gos. nacional'nyj issledovatel'skij un-t. Perm', 2022:40-45. URL: https://www.elibrary.ru/item.asp?id=49889579. (In Russ.).
11. *Aho A., Seti R., Ul'man D.* Kompilyatory: Principy, tekhnologii, instrumenty. M.: Vil'yams; 2008. 1184 p. (In Russ.).

Информация об авторах:

А. О. Корзников – бакалавр Пермского государственного национального исследовательского университета (614068, Россия, г. Пермь, ул. Букирева, 15), AuthorID 1206068;

Н. Н. Дацун – кандидат физико-математических наук, доцент кафедры математического обеспечения вычислительных систем Пермского государственного национального исследовательского университета (614068, Россия, г. Пермь, ул. Букирева, 15), AuthorID 734216.

Information about the authors:

A. O. Korznikov – BSc of Perm State University (15, Bukireva St., Perm, Russia, 614068), AuthorID 1206068;

N. N. Datsun – Candidate of Physical and Mathematical Sciences, Associate Professor of the Department of Mathematical Support of Computing Systems of Perm State University (15, Bukireva St., Perm, Russia, 614068), AuthorID 734216.