УДК 004.412:519.876.5

В.О. МИЩЕНКО

Харьковский национальный университет им. В.Н. Каразина, Украина

СОГЛАСОВАНИЕ ТРЕБОВАНИЙ ПРИ ФОРМАЛИЗАЦИИ ОДНОГО МЕТОДА ЭНЕРГЕТИЧЕСКОГО АНАЛИЗА ПРОГРАММ

Компьютерные программы обладают характеристиками, аналогичными работе и энергии физических систем. Впервые подобные характеристики вычислял М. Холстед, создатель «науки о программах». Большую роль в его подсчётах играла величина потенциального объёма. Но считалось, что его можно заменять мерой интеллектуального содержания программы. Это не так при энергетическом анализе программ, который имеет дело с многомодульными системами. Однако, один метод, использующий потенциальный объём, мог требовать субъективных решений при своей реализации. В статье синтезируется определение, которое позволяет ликвидировать этот последний источник субъективизма в измерении энергетических мер.

validation, program metrics, metrics desirable properties, program level, specification energy

Предыстория и актуальность проблемы

В этой работе речь пойдёт об обеспечении требований валидности [1] (validation) по отношению к некоторым определениям в системе мер (program metrics), которые отправляются от идей Холстеда [2]. Из шести свойств, требуемых в [1] от мер качества (metrics desirable properties) будут затронуты:

− осмысленность (meaningfulness) – наличие определения цели измерения и способа трактовки результатов измерения с точки зрения этой цели.

− корректность (correctness) – многогранное свойство, отражающее объективность процесса измерения, незаинтересованность измерителя в каком-то исходе, точность (определённость) результата.

− применимость (availability) – явно очерчена граница применимости.

Метрика потенциального объёма для программы или программного модуля определялась в [2] так:

$$V^* = (\eta_2^* + 2) \cdot \log_2(\eta_2^* + 2), \qquad (1)$$

где η^* – число различных параметров входа и выхода (а, точнее, классов, составленных из единых по смыслу фактически передаваемых параметров ([2] – стр. 31).

Через величину (1) в [2] определены: уровень программы (program level), работа (effort) и априорная оценка этой работы, послужившая в дальнейшем прообразом спецификационной энергии (specification energy) [3]. Некая субъективность, которая может присутствовать в расчёте значения (1) (при факторизации множества операций ввода-вывода), очевидно, стала причиной введения другой величины, получившей название интеллектуального содержания программы ([2] - стр. 47, 38):

$$I = \Lambda \cdot V, \qquad (2)$$

$$\Lambda = \frac{2\eta_2}{\eta_1 \cdot N_2}, \qquad (3)$$

где η_2 – число различных операндов, а N_2 – число всех операндов программы; η_1 – число различных операторов программы; V – холстедовский объём программы ([1, 2]).

Считалось [2], что вспомогательные меры (1) и (2) взаимозаменяемы, и при автоматизации подсчётов на практике использовалась (2) [1]. Данный вывод опирался на наблюдения за программным обеспечением 60-70-х годов прошлого века. Этот вывод, как мы объясним, не распространяется на модули современных программных систем. Поэтому в энергетическом анализе программ, который ориентирован на моделирование качества современных сис-

тем, была использована новая модификация метрики потенциальный объём [3]. Объективность определения числового атрибута этой меры зависит, во-первых, от объективности группировки операций входа-выхода, что удалось обеспечить, обойдя явную группировку [4 – 5]. После этого расчёт работы программирования по исходным текстам можно полностью автоматизировать.

Но потребность в автоматизации расчёта спецификационной энергии [3 – 5] делает актуальной разработку формального определения потенциального объёма для групп блоков схемы программной структуры (СПС). Блоки - это функции, процедуры и другие подобные конструкции, описанные в данном модуле (интерфейсном или подпрограмме верхнего уровня). По замыслу [3, 5] группировка блоков фактически закладывается разработчиком и должна обнаруживаться при анализе смысла модуля. Алгоритмизировать этот подход трудно [5].

Актуальным является такое обновление определения потенциального объёма групп блоков модуля, которое позволило бы автоматизировать расчёт спецификационной энергии. При этом необходимо обеспечить, как согласие с прежней неформальной трактовкой, так и учёт возможных альтернатив.

В данной работе за отправную точку были приняты высказанные в [5] идеи и алгоритм, испытание которых подсказало приемлемое по смыслу и удобству реализации корректное определение.

Цель и задачи работы

Языки программирования могут допускать внутри модулей программ различные конструкции (вроде пакетов, классов, задач), фактически объединяющие блоки (процедуры, функции и т. п.) в синтаксически оформленные группы. Например, для языка Ада это точно разъясняется в [5]. Поэтому вышесформулированная проблема группировки относится только к свободным блокам, то есть, лежащим вне синтаксически оформленных групп.

Цель данной работы состояла в такой доработке определений метрик потенциального объёма и спецификационной энергии на случай группировки свободных блоков модуля, чтобы сохранить осмысленность, применимость и обеспечить корректность.

Нашими задачами являются:

− рассмотрение возможностей подхода на базе расчёта интеллектуального содержания 4;

− формализация методов группировки, использующих комментарии и формальные параметры;

− сравнительное испытание этих методов на представительной группе тестов и использование результата для синтеза необходимого определения.

Интеллектуальное содержание?

Возможно обобщение метрики интеллектуального содержания на модули программ [5]. Чем она привлекательна? Корректностью. Расчёт требует только фильтрации структурных единиц текста программы. Проблема группировки блоков отпадает.

На чём основаны аргументы [5] в поддержку утверждения о близости величин (1) и (2)?

Каждый ввод данных подразумевает их обработку, а каждый канал вывода - наработку новых. Этим мотивируется выражение [2] уровня программы

$$\lambda = V/V^* \ , \qquad (4)$$

через объём программы V [2] и потенциальный объём (1) с тем, чтобы величина уровня характеризовала роль языка и метода в реализации программ. Другой способ учёта этих ролей показывает [2], что Λ (3) – это оценка уровня λ (4). Тогда из (2)-(4) вытекает, что I – это оценка V^*. По данным [2] на деле разность величин (1) и (2) относительно мала.

Однако для современных программных систем характерно не только большое число составляющих модулей, но и высокая степень их специализации, так что переданные в модуль данные не обязаны в нем обрабатываться (и наоборот), а введенные в описаниях операнды - в том же модуле использоваться! Поэтому осмысленность данной метрики,

если не теряется вообще, то заведомо изменяется, теряя явную связь с потенциальным объёмом.

Другой аргумент. Возможны три ситуации. Первая – программы нет вовсе, но доступна специфицирующая её документация. Тогда потенциальный объём можно вычислить, а интеллектуальное содержание – нет. Аналогично, если: разработаны только интерфейсные модули и блоки. И только в случае готовой программы вычислимы обе меры.

Подведём итог, учтя также и аргументы [5]:

Замена потенциального объёма интеллектуальным содержанием заменяет проблему сложности подсчёта входов-выходов проблемой точного определения операторов и операндов, что совсем не тривиально (например, [6]).

Метрика (2) по сравнению с (1) имеет более узкую область применимости. Больше того, она выхолащивается для классов тех модулей, для которых характерна стабильность соотношения между операторами и операндами (и их алфавитами) [5].

Итак, для модулей больших систем взаимозаменяемость обсуждаемых метрик не только не обоснована, но в ряде случаев легко оспаривается.

Группировка свободных блоков

В соответствии с используемыми нами определениями [3 – 5] потенциальный объём модуля СПС образуется как сумма потенциальных объёмов (1) тех его блоков, которые не являются телами (реализациями) других блоков. Эта величина не зависит от группировки, но так используется лишь при подсчёте работы программирования модуля. Другая основная мера энергетического анализа – спецификационная энергия E подсчитывается для модулей суммированием, существенно используя V^* групп [3]! Для группы, исключая автономную группу модуля и редкие усложнённые группы:

$$E = \overline{\lambda}^{-2} \cdot \left(\sum_i V_i^* \right)^3 \quad \text{(сумма по всем блокам)}, \quad (5)$$

где $\overline{\lambda}$ - уровень языка [2]: - средняя величина уровней (4), подсчитанная по широкому классу модулей, в который входит данный модуль (такие классы характеризуется однотипным использованием данного языка программирования), V_i^* - потенциальный объём вида (1) i -го блока.

Нелинейность зависимости от V_i^* делает спецификационную энергию весьма нестабильной при росте потенциальных объёмов групп (по сравнению с этим можно пренебречь проблемами, которые связаны с точным определением величины $\overline{\lambda}$). Если никогда не производить группировку, то уже при наличии в модуле нескольких десятков блоков, имеющих $\eta^* \sim 10$ (что, по опыту можно сказать, немого), величина энергии модуля окажется на несколько порядков больше работы программирования. Если, напротив, из каждого блока всегда составлять отдельную «группу», эффект будет противоположным. Меж тем, основное предположение энергетического анализа состоит в том, что разность энергии и работы для разных модулей (и разных версий одного модуля) может иметь произвольный знак, определяемый соотношением между трудностью спецификации программистской задачи и затратностью кодирования. Так будет, если учесть группировку блоков по смыслу их разработки.

Всюду в данной работе предполагается, что рассматриваемый код разработан программистом-человеком. Это означает, что как бы ни был велик размер модуля программы, разработчик в каждый момент держит в уме лишь некоторую осмысленную часть структурированного кода. В соответствии с известным психологическим законом «7±2» [7] следует ожидать, что при наличии хотя бы более чем пол десятка свободных блоков, разработчик явно или подсознательно разбивает их на группы числом, по возможности, не более 7+2. При этом, за редкими исключениями, численность групп, по всей вероятности, должна составлять 7±2 блоков.

Рассмотрим самостоятельные модули [3, 5] и в

них те блоки, которые не отнесены к обязательным группам по синтаксическим признакам. Например, в языке Ада это те, которые не объединены внутренними пакетами или задачами.

Определение 1. При отсутствии иных указаний со стороны разработчика (эксперта) следует относить к автономной группе только остаточный блок, остатки группирующих синтаксических конструкций и все описания констант, считая их, в отличие от описаний переменных, вырожденными блоками без тел и параметров входа-выхода.

Это определение, в частности, важно тогда, когда модуль содержит сотни описаний констант. Нужно иметь в виду, что для автономной группы

$$E = \sum_i \overline{\lambda}^{-2} \cdot V_i^* \text{ (сумма по блокам).} \qquad (6)$$

Определение 2 (вариант 1). Серии комментариев, занимающих отдельные строки, считать разделителями меду блоками разных групп. Для таких группирующих комментариев желательны дополнительные признаки (но это зависит от разработчика).

Следующий вариант получен доработкой (с учётом испытания на модулях реальных программ) алгоритма, предложенного в [5] (стр. 113-114).

Определение 2 (вариант 2). Свободные блоки (именуемые здесь «блоки») группировать так.

Г-0. Множество блоков подвергнем трём различным *факторизациям*: *слабой*, *нормальной*, *строгой*. Строгая факторизация отождествляет блоки, у которых одинаковы последовательности имен типов (в языке Ада – подтипов) формальных параметров, причём виды соответствующих параметров должны совпадать (результат функции, если есть, понимается как последний параметр вида «выход»). Нормальная факторизация исходит из строгой. Она объединяет те её классы, которым соответствуют последовательности имён подтипов параметров, совпадающие с точностью до перестановки, а также те, которым соответствуют последовательности, куда в любом числе входит одно одинаковое для всех последовательностей имя (виды параметров не учитываются). Слабая факторизация вообще отождествляет все те блоки, которые имеют одинаковые множества различных имён типов (подтипов) формальных параметров. Данные факторизации вложены одна в другую, и для чисел их классов имеем:

$$s_{slight} \le s_{norm} \le s_{strong} \qquad .(7)$$

Г-1. Пусть блоков не менее $7(k-1)$, но менее $7k$ ($k \ge 1$). Тогда:

1) если $s_{strong} < k-1$, то основать группировку на строгой факторизации;

2) если в ряду чисел (7) нет k, но есть $k-1$, то за основу группировки блоков на данном этапе принять факторизацию, которой соответствует вхождение $k-1$ в ряд (7) (если таких вхождений несколько, то соответствующие факторизации совпадают);

3) если в ряд чисел (7) входит k, то за основу группировки блоков на данном этапе принять факторизацию, которой соответствует это вхождение;

4) если среди чисел (7) нет ни k, ни $k-1$, но для одного из входящих туда чисел s верно, что $s \le k-1$, то за основу группировки принять факторизацию, которой соответствует вхождение наибольшего из таких s;

5) если $k < s_{slight}$, за основу группировки блоков принять слабую факторизацию;

Г-2. Если на каком-то этапе группировки все классы блоков содержат не более 7+2 элементов, то закончить, иначе применять к ним рекурсивно Г-1. с выходом для классов, где $s_{strong} = 1$.

Г-3. По окончании должно быть не более 7×7 групп блоков, среди которых нет таких, что

$$V^* > 250 . \qquad (8)$$

Иначе результат сомнителен, требуется экспертиза.

Г-4. Группы блоков, состоящие из 1 блока, объединить с автономной группой, а к остальным группам свободных блоков применять (4).

Испытание на стандартных модулях

Рассматривались библиотечные модули, описанные в приложении А к руководству по языку Ада 95 [8], поскольку они отражают особенности нескольких разных прикладных областей и могут рассматриваться, как примеры канонического стиля программирования. Исключив «тривиальные» пакеты и те описания, которые схематически аналогичны рассмотренным, приведём данные по следующим модулям (в скобках указано число блоков, наличие операций ввода-вывода в теле, если они есть, и число близких аналогов этого модуля в приложении А):

1. Ada.Storage_IO (2, ввод-вывод);
2. Ada.Command_Line (4, ввод-вывод);
3. Ada.Numerics.Float_Random (7, есть1 аналог);
4. Ada.Sequential_IO (13, ввод-вывод, 2 аналога);
5. Ada.Strings.Maps (18, есть 1 аналог);
6. Ada.Characters.Handling (26);
7. Ada.Numerics.Generic_Elementary_Functions (29);
8. Ada.Strings.Fixed (31);
9. Ada.Strings.Fixed.Generic_Bounded_Length (67, имеется 1 аналог);
10. Ada.Text_IO (77, ввод-вывод, имеется 1 аналог);

Были испытаны варианты 1 и 2 определения 2. Критерий сомнительности (8) был принят также для групп, образованных по варианту 1. В качестве числового индикатора качества группировки использовалась часть спецификационной энергии модуля, приходящаяся на его свободные блоки. При наличии ввода-вывода и p формальных параметров блока для него полагалось $\eta_2^* = 2p$, имея в виду операции, скрытые в теле. Значение уровня языка бралось

$$\overline{\lambda} = 1.5 .$$

Результаты помещены в табл. 1, где используются сокращения:, «ч.гр.» - число групп, «kHd» это 1000 Hd или 1_000_000 бит-симв. Отметим, что относительная вариация (отношение модуля разности к полусумме) в худшем случае имеет значение 2.

Расхождение результатов оценивания E на порядок показывает, что, по крайней мере, один из вариантов неадекватен. Это вариант 2, поскольку он завышает число групп (обычно в них меньше 5 блоков). Но 1-й вариант, очень близкий к группировке по смыслу, обнаружил тенденцию к ограничению числа групп в ущерб ограничению численности групп, провоцируя (8). Но главный недостаток – нет гарантий наличия нужных комментариев.

Таблица 1

Эффект методов группировки по вариантам 1, 2

№	Указание комментариями		Анализ параметров		Отн. вариация E
	ч.гр.	вклад в E (kHd)	ч.гр.	вклад в E (kHd)	
1	1	0,013	1	0,013	0,0
2	1	0,027	3	0,0024	1,67
3	2	0,0185	4	0,0044	1,23
4	2	2,24	7	0,168	1,72
5	3	0,680	8	0,075	1,60
6	4	0,378	11	0,305	0,21
7	1*	10,08*	3	1,83	1,39
8	6	3,30	24	0,050	1,94
9	6*	35,18*	27	0,92	1,90
	7	9,19	17	2,89	1,04

* сомнительно, т. к. для одной из групп верно (8)

Для тестирования 3-го варианта мы для всех рассматриваемых модулей примем за ориентировочное приближение к E значение первого варианта по табл. 1, кроме модулей № 7, 9. Для тех возьмём геометрические средние по двум вариантам.

Моделирование группировки

Ввиду неудовлетворительности рассмотренных методов, откажемся от явной группировки. Смоделируем осреднённый эффект «рациональной» группировки. Исходим из того, что для модулей с числом блоков $n = 1..4$ она не нужна, при $7 - 2 \le n \le 7 + 2$ образуется 1 группа с вероятностью близкой, но не равной 1, а при больших n образуются группы с числом блоков, примерно равным 9.

Определение 2 (вариант 3). Для модуля с n свободными блоками их вклад в E модуля составляет

$$E_f = \begin{cases} \left(V^*\right)^3 \cdot \lambda^{-2} & \Leftarrow n \le 4 \\ (n/9+0.5) \cdot \left(V^*/(n/9+0.5)\right)^3 \cdot \lambda^{-2} & \Leftarrow n \ge 5 \end{cases} \quad (9)$$

где V^* - сумма потенциальных объёмов n блоков.

Таблица 2

Моделирование среднего эффекта группировки

№ мод.	V^*	Число блоков	E (kHd)	Отн. вариация E
1	31,02	2	0,013	0,0
2	39,51	4	0,027	0,0
3	52,75	7	0,040	1,29
4	218,5	13	1,226	0,59
5	176,5	18	0,391	0,54
6	222,4	26	0,426	0,12
7	283,1	29	0,728	1,42
8	321,5	31	0,949	1,11
9	859,2	67	4,466	0,24
10	837,7	77	3,186	0,97

Результаты по данному варианту (табл. 2) для 9-ти тестов меньше ориентира E из предыдущего раздела, но не больше, чем в 2-3 раза (кроме № 6, где почти равенство и № 7, где больше в 6 раз). Это согласуется с отмеченной выше тенденцией к завышению E в варианте с использованием комментариев и тенденцией к занижению в варианте 3, ориентированном на численное равенство групп. Для данного варианта определения свойство применимости практически не ограничено, осмысленность сохраняется, корректность гарантированна.

Заключение

Определение метрик энергетического анализа на основе группировки блоков свободных модулей сталкивается с конкуренцией между осмысленностью и корректностью. В работе эта трудность обходится за счёт отказа от явной группировки в пользу моделирования среднего эффекта от виртуальной группировки. Это приводит к формальному определению спецификационной энергии модулей программ, которое не требует субъективных суждений.

Теперь можно строить компьютерные приложения для автоматизации расчёта указанной метрики по исходным текстам программ.

Литература

1. ISO/IEC TR 9126 3:2003 [Электр. ресурс]. – Режим доступа: http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=22891.

2. Холстед М.Х. Начала науки о программах. – М.: Финансы и статистика, 1981. – 128 с.

3. Мищенко В.О. Математическая модель стиля Software Science для метрического анализа сложных наукоёмких программ // Вісник Харк. нац. ун-ту. Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». – 2004. – № 629, вип. 3. – С. 70-85.

4. Mishchenko V.O. One experiment in using energy metrics proposed for software process assessment // Radio-electronic and computer systems. – 2007. – № 8. – P. 121-124.

5. Мищенко В.О. Энергетический анализ программного обеспечения с примерами реализациии для Ада-программ. – Х.: ХНУ им. В.Н. Каразина. – 2007. – 119 с.

6. 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. – Vol. 22, Issue 5. – ISSN:0362-1340. – P. 32-41.

7. Столяренко Л.Д. Основы психологии. – Ростов н/Д: Феникс, 2001. – 655 с.

8. Ada Reference Manual. Language and Standard Libraries. Version 6.0. 21 December 1994. – Intermetrics, Inc. – 1995. – 569 p.

Поступила в редакцию 12.02.2008

Рецензент: д-р техн. наук, проф. В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.