УДК 004.05+519.6

Д.Н. ЛИТВИНОВ, В.О. МИЩЕНКО

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

ПРИМЕНЕНИЕ ЭНЕРГЕТИЧЕСКИХ МЕТРИК ДЛЯ ОЦЕНКИ ИСПОЛЬЗОВАНИЯ ASIS И В ДРУГИХ ПОДОБНЫХ ЗАДАЧАХ

Каким образом обосновать относительную целесообразность использования тех или иных инструментальных библиотек при разработке программных систем? Если не проводилось строгих исследований по обобщению чужого опыта, да ещё и требования к разрабатываемой системе не типичны, то ответ получить не просто. В настоящей статье рассматривается один аспект затронутой проблемы, состоящий в экономии усилий разработчиков. Мы формулируем, аргументируем и испытываем на конкретном примере метод сравнения трудоёмкости выполнения различных программистских проектов, который может быть использован для получения ответа на указанный вопрос. Метод использует систему недавно разработанных энергетических метрик.

Ключевые слова: *инструментальная библиотека, программная система, трудность, метрики, исходный код, энергия, работа.*

Введение

Использование всевозможных библиотек поддержки прикладного и системного программирования давно стало одним из важнейших инструментов разработки программного обеспечения. Вопрос об их продуктивности часто не стоит, поскольку новое инструментальное средство такого рода может до поры до времени не иметь конкурентов, а большинство потенциальных пользователей – квалификации для самостоятельной реализации тех функций, поддержку которых такая библиотека обеспечивает. В тех же важных случаях, когда альтернатива есть, научно обоснованные методы выбора между альтернативами мало распространены в широкой практике. В многочисленных проектах решения принимаются разработчиком на основании субъективных суждений относительно имеющегося у него опыта (если он есть), прикидочной оценки возможных выгод и рисков, а также рекламы и опубликованных высказываний экспертов (что, не редко, может быть одно и то же). Поэтому разработка объективных методов оценки сравнительного эффекта применения инструментальных библиотек поддержки программирования, полезных для широких категорий разработчиков, является актуальной задачей. На чём могут основываться такие методы, если они, прежде всего, ориентированы на разработчиков со скромными бюджетами?

Один из популярных современных стандартов, ISO 9126 [1], который совместно с другими стандартами, включая [2], обеспечивает основу определения качества программной продукции при закупке и поставке, описывает технологию и организацию измерений и анализа, позволяющие давать в отношении различных сторон качества программных систем заслуживающую доверия количественную оценку.

Известный документ [3] обеспечивает стандартизацию техники метрических измерений программной продукции и процесса её производства, а также обзор подходящих метрик, известных к концу 80-х гг. прошлого века.

Мы развиваем подход к оценке инструментальных библиотек, состоящий в том, чтобы увязать оценивание их качества в использовании по аспекту продуктивности [1] с оценкой преимуществ применения в конкретном процессе разработки [3]. Необходимыми для реализации такого подхода свойствами обладают энергетические метрики [4], разработка которых отталкивалась от богатого коллективного опыта работы с научными метриками (по поводу которого см., например, [3-5]).

Испытание разработанного нами метода демонстрируется на примере важной задачи о сравнительной оценке преимуществ использования библиотеки реализации Ada Semantic Interface Specification (ASIS), которая является частью стандартного окружения языка программирования Ада [6 – 8].

1. Постановка задачи исследования

Цель данной работы состояла в том, чтобы, задавшись подходящим критерием продуктивности для оценки инструментальных библиотек, выработ-

тать и обосновать метод сравнительной оценки двух или нескольких библиотек на основании метрического анализа результатов их использования в разных, вообще говоря, проектах.

Для этого решались следующие задачи:

Т: выработка требований к мерам эффективности использования разных библиотек;

М: обоснование выбора системы метрик для осуществления оценки усилий программирования в разных проектах;

С: разработка критерия и метода оценивания продуктивности использования библиотек;

П: испытание на примере решения практической задачи оценки использования библиотеки АСИС.

2. Система измерений

Решение задачи М. Напомним, что «Метрики качества в использовании замеряют степень, в которой продукция удовлетворяет запросам оговоренной категории пользователей по достижению предписанных целей эффективно, продуктивно, безопасно и с максимальной степенью удовлетворенности при оговоренных условиях использования» [1]. Безопасность и чувство удовлетворённости пользователя иногда играют очень важную роль, но далеко не всегда.

С другой стороны, очевидно, что эффективность исследуемой программной библиотеки с точки зрения пользователя каким-то образом и так устанавливается каждым пользователем, раз уж он рассматривает вопрос о выборе данной библиотеки против других альтернатив. Поэтому решающей характеристикой, определяющей этот выбор, и тем самым, привлекающей наше внимание в данном исследовании, оказывается продуктивность в использовании. Однако задача её проверки стоит здесь иначе, чем в [1].

Действительно, там речь идёт о выработке абсолютного среднего значения по результатам тестирования одного продукта многими пользователями по многим аспектам. А преимущества использования библиотеки – это относительный показатель при рассмотрении конкретных альтернатив. При этом, по определению продуктивности, значение показателя эффективности (его выбору в данной работе посвящена задача *Т*) делится на меру, которая должна отражать затраты ресурсов. По этому поводу в [1] отмечается: «Наиболее общим ресурсом является время выполнения задания [системой, в состав которой входит изучаемая продукция], тогда как другими подходящими для рассмотрения ресурсами могли бы быть усилия пользователя, расход материалов или финансовые расходы в связи с использованием». Если вид продукции – инструментальные библиотеки, а пользователь – разработчик собственных программ, который такими библиотеками пользуется, то роль «времени выполнения» для разных программ зависит от их назначения и может колебаться от ключевой до совершенно не значимой. Тем самым, рискованно избирать её в качестве универсального вида ресурса при всех сравнениях. Более универсальным в этом контексте представляется ресурс усилий разработчиков. Примем, что именно отражающая его мера должна попадать в знаменатель отношения, выражающего продуктивность при наших оценках.

Имея в виду выбор такой меры процесса, которая, как мы увидим, не может рассматриваться изолированно от других характеристик, предъявим к интересующей нас системе метрик вполне естественные требования:

- абстрактность (чем формальнее трактовка метрики, тем больше шансов, что она с одинаковым успехом окажется применимой в отношении самых разных программных библиотек);
- наглядная интерпретируемость (абстрактность меры антагонистична её непосредственной наглядности, но может существовать интерпретация, не обязательно всегда чёткая и точная, но достаточно наглядная для того, чтобы подсказывать конкретные выводы);
- комплексность (необходимы и меры, которые интерпретируются как общие затраты на разработку программы, и такие, которые служат индикаторами её надёжности);
- как можно большая детерминированность;
- пригодность к автоматизации измерений;
- минимально низкая стоимость, как внедрения в практику менеджмента, так и отдельных измерений.

Если сравнивать с данными требованиями те виды метрик, которые описаны в [1], то наиболее близки к нашим требованиям метрики науки о программах (М. Холстеда), которые короче именуют «научными». Они отличаются тем, что полностью удовлетворяют второму, третьему и последнему требованиям, в большой степени - первому (абстрактность), но недостаточно детерминированы, что препятствует автоматизируемости оценки. Не вдаваясь в эти проблемы, достаточно полно освещённые, например, в [9, 4], особо отметим не строгий характер определения некоторых примитивных параметров.

Поэтому с «теоретическими» научными метриками сосуществуют научные метрики, которые трактуются, как их «оценки» [5, 9]. В нашей работе используется в первозданном виде только одна из

научных метрик – *оценка трудности* (компонентов программы) [3]:

$$\hat{D} = (n_1 / n_1^*) \cdot (N_2 / n_2) \quad , \qquad (1)$$

где N_2 – число входящих в исходный текст компонента программных символов (грубо говоря, лексем), которые трактуются как операнды;

n_2 – словарь операндов;

n_1 – словарь операторов;

$n_1^* = 2$ – словарь операторов при реализации на потенциальном языке (см., напр., [4, С. 10-11]).

Эмпирически установлено, что безразмерная величина $\hat{D}$ является индикатором «склонности к ошибкам» [5]. Правда, пороговые значения этой метрики, о которых сообщается в [3, 5] и других доступных нам источниках, указаны там для языка ПЛ/1. Огрубив их для надёжности примерно на 20%, будем считать, что программный компонент с трудностью меньшей 100 заслуживает доверия, больше – требует внимания, а трудность свыше 190 опасна.

На место «наследника» науки о программах претендует энергетический анализ [4]. Он обобщил важнейшие научные метрики, добавил к ним спецификационную энергию E (аналог внутренней энергии термодинамических систем), объём W разработки компонента программы, который учитывает в себе и объём по Холстеду, и дополнительный объём, связанный с использованием ресурсов программирования из других компонент [4], и строго определил трудность D. В частности, теперь метрика работы

$$A = D \cdot W \qquad (2)$$

для модуля (компонента). Её абстрактность повысилась, поскольку её можно корректно суммировать по модулям систем и подсистем. Аналог первого начала

$$E = A + Q \qquad (3)$$

вводит метрику Q - интеллектуальное тепло (аналог тепла в термодинамике) [4].

Метрики E, A, Q и все с ними связанные называются энергетическими. Они вполне детерминированы. Практическим доказательством автоматизируемости их оценивания служат приложения SS_Ada_Scanner [4] и EE_Ada_Struc (о котором здесь сообщается впервые).

Первое из них, разработанное вторым из авторов статьи, в текущей версии 2011 г. оценивает объём разработки программных систем, разработанных на языке Ада 95 (может перенастраиваться для Ады 2005 и 2012), а также основные метрики науки о программах (SS в названии – от Software Science).

Второе, созданное первым из авторов, подсчитывает потенциальные объёмы модулей, которые необходимы для расчёта D, и спецификационную энергию для программ, разработанных на Ада 95 (допустимость следующих стандартов зависит от согласованности с обновлениями ASIS). После выполнения этих приложений работа и тепло рассчитывается простейшей утилитой.

Итак, метрика (2) и связанные с ней другие энергетические метрики, удовлетворяют всем сформулированным выше требованиям и могут использоваться при оценке продуктивности библиотек.

Решение задачи T. Метрики эффективности в использовании «... регистрируют, может ли пользователь точно и полностью достичь своих целей в оговоренном контексте использования» [1]. Соответствующие меры уточняются для каждого конкретного случая (приводимые в [1] метрики являются схемами, которые становятся определёнными мерами после конкретизации таких понятий, как «задачи пользователя», «завершение задачи» и т. п.). Поэтому мы ограничимся следующими довольно общими условиями, которые нужно выполнить, подбирая меру эффективности при оценивании библиотек:

это должна быть экстенсивная величина T, которая тем больше, чем больше полезной для пользователя-программиста работы в состоянии выполнять его программа;

числовое значение $T > 0$ данной меры должно быть увязано с некоторыми признаками программы пользователя $P_1, \ldots P_k > 0$ так, чтобы можно было принять гипотезу:

$$T \approx c \cdot P_1^{1/k} \cdot \ldots P_k^{1/k}, \qquad (4)$$

где $c > 0$ – константа (не обязательно оценённая), общая для всех программ интересующего пользователя класса C.

3. Метод сравнительной оценки инструментальных библиотек

Решение задачи C. Ряд программных библиотек, имеющих общую область приложения, будем сравнивать относительно некоторого класса программ C попарно. Предполагается, что использование этих библиотек в разработках для этого класса эффективно, и для них справедлива гипотеза (4) с одними и теми же постоянными. Пусть каждая из библиотек нашла характерное применение в некотором проекте, а все эти проекты, вообще говоря, разные. Если какая-то из библиотек применялась в нескольких проектах, которые все одинаково показательны, то потом можно будет выполнить осреднение результатов по этим проектам. Поэтому счита-

ем, что для каждой библиотеки рассматривается по одному проекту.

Продуктивность библиотеки полагается равной

$$\Pi = T/A, \qquad (5)$$

где T – мера (4) эффективности библиотеки на разработанной программе пользователя;

A – суммарная по всем компонентам этой программы работа программирования (2).

Из двух библиотек та лучше по критерию продуктивности (5), для которой эта величина больше.

Поэтому в каждом из рассматриваемых проектов нужно оценить работу программирования A и параметры продуктивности T. Однако для корректности сравнения работ требуется, чтобы исходные тексты сравниваемых программ свидетельствовали о сходности идеальных процессов, которые могли бы привести к созданию этих программ при оптимальном течении разработки. Следующие признаки свидетельствуют о такой сходности:

Признак 1. Соизмеримая трудность компонент: предпочтительно, чтобы все компоненты сравниваемых программ имели оценку трудности $\hat{D} < 190$ или процент исключений был примерно одинаков.

Признак 2. Одинаковая энергетическая сбалансированность: для сравниваемых программ сложность задачи программирования, которую представляют энергии $E_{1,2}$, и суммарные по всем компонентам идеальные затраты, представленные работами $A_{1,2}$, должны быть такими, чтобы

$$\varepsilon \cdot (E_2 / A_2) \le E_1 / A_1 \le \varepsilon^{-1} \cdot (E_2 / A_2), \qquad (6)$$

где $\varepsilon = 0{,}1$ – ориентировочно, по нашему опыту.

Для пары сравниваемых библиотек определим

$$I_j = P_j^{(2)} / P_j^{(1)} \ (j=1..k), \qquad (7)$$

где $P_j^{(i)}$ – j–й признак эффективности (см. (4)) для i–й из двух программ.

Иногда, если точная оценка какого-то признака затруднена, то вместо вычисления I_j по формуле (7) может оказаться проще оценить сам этот индекс.

Оценка продуктивности второй библиотеки относительно первой определяется по формуле:

$$\pi_{2,1} = \Pi_2 / \Pi_1 = I_1^{1/k} \cdot \ldots I_k^{1/k} \cdot A_1 / A_2 . \qquad (8)$$

Отметим два свойства этой метрики. *Комплексность*: если выбранную меру эффективности T оценивать одним признаком, но проведено несколько предварительных сравнений пары библиотек с использованием разных признаков, то, введя комплексную оценку (4) со всеми этими признаками, можно получить значение относительной продуктивности (8) как среднее геометрическое предыдущих оценок. *Дифференцируемость*: если роль разных признаков по отношению к выбранной мере T не равнозначна, то это учитывается заменой в (4), (8) степеней $1/k$ подходящими степенями λ_j так, чтобы $\lambda_1 + \ldots \lambda_k = 1$.

4. Актуальный пример

Решение задачи П. ASIS позволяет выполнять из прикладных, инструментальных или системных программ стандартизованные запросы относительно любых доступных откомпилированных Ада программ в терминах руководства по этому языку. Очевидно, что многие задачи статического анализа программных систем, разработанных на языке Ада, и задачи автоматизации манипулирования исходными текстами на этом языке с различными программистскими целями имеют в лице ASIS высокоуровневое средство решения. Однако использование в программе ASIS исключает её конверсию на другой язык программирования или создание аналогичной на другом языке.

Мы исследовали, имеется ли преимущество от использования ASIS при разработке программ для оценки метрик по исходным текстам Ада программ. Проводить прямые эксперименты на этот предмет с параллельными разработками при помощи разных библиотек было невозможно ввиду ограниченности ресурсов. Действительно, такие программы, точно выполняющие оценки содержательных метрик, должны при синтаксическом разборе исходного кода учитывать нетривиальные контекстные условия. Они довольно сложны, трудоёмки, и требуют значительного времени на доведение качества в использовании до необходимого уровня. Поэтому нами был применён метод раздела 2.

Мерой эффективности будет служить работа по синтаксическому анализу, связанная с признаками:

P_1 – число узлов в синтаксическом дереве, в соответствии с которым анализируется исходный текст Ада программы (это семантический признак);

P_2 – число вызовов методов используемой библиотеки по всем исходникам программной системы (это признак синтаксический).

В качестве первой библиотеки выступала реализация ASIS для GNAT AdaCore.

В качестве альтернативной библиотеки более низкого уровня выступила система на базе GNAT.Spitbol.Patterns (из системы поддержки программирования GNAT AdaCore) вместе с ориентированными на неё модулями:

SS_Ada_Symbols_Check,

SS_Words_Identification,

SS_Tables,

разработанными в рамках прикладной системы SS_Ada_Scanner.

Испытательный проект с номером 1 – разработка EE_Ada_Struc, с номером 2 – SS_Ada_Scanner.

В составе продукции проекта №2 29% модулей имеют опасную трудность (большую 190), а их средняя трудность 156 высока, но не опасна и относительно принятого нами порога (190), и относительно известного критического порога для PL/1 (160). Отношение E/A = 0.026 (что далеко от 1.0 и свидетельствует о трудности реализации проекта). В проекте №1 EE_Ada_Struc первоначально половина модулей оказались опасно трудными, их средняя трудность 387, а E/A = 0,006. Очевидно, оба признака сходности процессов из раздела 2 были нарушены. Причину проблем проекта №1 легко усмотреть в том, что в двух из 4-х его компилируемых модулях сосредоточено много тел внутренних подпрограмм. Используя стандартные возможности языка Ада, мы получили другое представление этой программы путём механического переноса этих тел в отдельно компилируемые субмодули. После этого процент опасно трудных модулей сократился до 30.8%, средняя трудность по модулям составила 92, а E/A = 0,205. Оба признака сходности теперь, очевидно, выполнены. Можно продолжать анализ. Данные измерений параметров эффективности и работы приведены в табл. 1.

Таблица 1

Признаки эффективности
и работа программирования для программ,
использующих разные библиотеки

	1. EE_Ada_Struc	2. SS_Ada_Scanner
P1	52	107
P2	37	64
A	$5{,}0\cdot10^6$	$20{,}0\cdot10^6$

Итоговый результат:

$$\pi_{2,1} = \Pi_2 / \Pi_1 = 2,2 \qquad (9)$$

подтверждает предполагавшееся техническое преимущество использования ASIS. Для равноэффективных программ рассматриваемого класса библиотека, которая поддерживает ASIS, позволяет вдвое экономить расчётную работу программирования по сравнению с библиотекой, которая опирается на возможности Снобола, средства которого реализует GNAT.Spitbol.Patterns (однако в случае проекта №2 использование ASIS исключалось требованием модифицируемости с целью оценки программ на языках, отличных от Ады).

Выводы

На базе современной теории метрической оценки программного обеспечения разработан метод сравнительной оценки продуктивности использования инструментальных библиотек программирования.

Его применение позволило дать оценку использования ASIS в Ада, которая может рассматриваться как ориентир степени повышения производительности программирования.

В дальнейшем имеет смысл исследовать эффект использования ASIS в других классах программ и развить разработанный здесь метод оценки, выбирая другие меры эффективности.

Литература

1. ISO/IEC 9126-4:2004 [Text]. – Software engineering – Product quality – Part 4: Quality in use metrics.

2. ISO/IEC 15504-1:2004 [Text].– Information technology – Process assessment – Part 1: Concepts.

3. IEEE Std 982.2-1988. IEEE Guide for the Use of Standard Dictionary of Measures to Produce Reliable Software [Электронный ресурс]. – Режим доступа: http://members.aol.com/geshome/IEEE982/IEEE9822.pdf. – 15.01.2012 г.

4. Мищенко, В.О. Энергетический анализ программного обеспечения с примерами реализации для Ада-программ [Текст] / В.О. Мищенко. – Х.: ХНУ имени В.Н. Каразина, 2007. – 119 с.

5. Shen, V.Y. Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support [Text]/ V.Y. Shen, S.D. Conte, H.E. Dunsmore // IEEE Transactions on Software Engineering. – Mar 1983. – Vol. SE-9, no 2. – P 155 – 165.

6. ISO/IEC 15291:1999 Information technology – Programming languages – Ada Semantic Interface Specification (ASIS) [Text].

7. Introduction to ASIS. – [Электронный ресурс]. – Режим доступа: http://www.sigada.org/wg/asiswg/intro.html. – 15.01.2012 г.

8. ASIS. Ada Semantic Interface Specification. – [Электронный ресурс]. – Режим доступа: http://www.adacore.com/home/products/gnatpro/add-on_technologies/asis. – 15.01.2012 г.

9. Hamer, P.G. Halstead's Software Science – A Critical Examination [Text]/ P.G. Hamer, G.D. Frewin // Proc. of the 6th International Conference on Software Engineering. – Tokyo, Japan, Sep 13-16, 1982. – P. 197-206.

Поступила в редакцию 6.03.2012

Рецензент: д-р техн. наук, проф. А.Л. Ляхов, Полтавский национальный технический университет им. Юрия Кондратюка, Полтава, Украина.

ЗАСТОСУВАННЯ ЕНЕРГЕТИЧНИХ МЕТРИК ДЛЯ ОЦІНКИ ВИКОРИСТАННЯ ASIS І В ІНШИХ ПОДІБНИХ ЗАДАЧАХ

Д.М. Литвинов, В.О. Міщенко

Як обгрунтувати доцільність використання тих чи інших інструментальних бібліотек при розробці власних програмних систем? Якщо не проводилося строгих досліджень з узагальнення чужого досвіду, та ще й вимоги до розроблюваної системі не типові, то відповідь отримати не просто. У цій статті розглядається один аспект порушеної проблеми, що складається в економії зусиль розробників. Ми формулюємо, аргументуємо та випробуємо на конкретному прикладі метод порівняння трудомісткості виконання різних програмних проектів, який може бути використаний для отримання відповіді на зазначене питання. Метод використовує нещодавно побудовану енергетичну модель програмних систем.

Ключові слова: інструментальна бібліотека, програмна система, трудність, вихідний код, енергія, робота.

APPLYING ENERGETIC METRICS TO EVALUATION OF USING ASIS AND OTHER SIMILAR TASKS

D.N. Litvinov, V.O. Mishchenko

How can we evaluate the feasibility of using certain instrumental libraries in our software projects? If there are no rigorous research results on generalization the experience of others, and even requirements for the system being developed are not typical – then we hardly can get the answer. In this paper we consider one aspect of this problem: saving development effort. We formulate, argue, and evaluate on concrete example the method of comparing development effort of software projects. This method can be used to answer the question stated above. We use the energetic metric for software systems proposed recently.

Key words: toolkit library, software system, difficulty, source code, energy, effort.

Литвинов Дмитрий Николаевич – младший научный сотрудник кафедры моделирования систем и технологий Харьковского национального университета им. В.Н. Каразина, Харьков, Украина.

Мищенко Виктор Олегович – канд. физ.-мат. наук, доц., доц. каф. моделирования систем и технологий Харьковского национального университета им. В.Н. Каразина, Харьков, Украина.