розроблених поведінкових моделях усіх особливостей функціонування троянських програм в комп'ютерних систем шляхом порівняння реальних шкідливих програмних об'єктів із розробленими моделями.

Розроблено модель процесу діагностування комп'ютерних систем на наявність троянських програм, яка базується на залученні компонентів штучного інтелекту, зокрема нечіткої логіки та алгоритмів штучних імунних систем, і відрізняється від відомих тим, що використовує поведінкові моделі класів троянських програм, дозволяє адаптувати процес діагностування до окремо взятої комп'ютерної системи і не потребує побудови баз сигнатур.

Література

1. Савенко О. Дослідження методів антивірусного діагностування комп'ютерних мереж / Олег Савенко, Сергій Лисенко // Вісник Хмельницького національного університету. – 2007. – № 2. – Т. 2. – С. 120–126.

2. Williamson M. M. Virus throttling / M. M. Williamson, J. Twycross, J. Griffin, and A. Norman // Virus Bulletin. – 2009.

3. Proactive/Retrospective test. Anti-Virus comparative [Електронний ресурс]. – Режим доступу : http://av-comparatives.org.

4. Савенко О. Модель процесу пошуку троянських програм в персональному комп'ютері / Олег Савенко, Сергій Лисенко // Радіоелектронні і комп'ютерні системи. – 2008. – № 7. – С. 87–92.

5. Савенко О.С. Поведінкова модель троянських програм / О.С.Савенко, С.М. Лисенко // Комп'ютерні науки та інформаційні технології (CSIT-2007): міжнар. наук.-техн. конф., 27–29 вересня 2007 р. : тези доповідей. – 2007. – С. 129–132.

6. Система пошуку троянських програм з використанням нечіткого логічного висновку: зб. наук. праць міжнародної науково-практичної конференції [«Інтелектуальний аналіз інформації IAI-2008»], (Київ, 14-17 травня 2008 р.) / [редкол.: С.В. Сирота та ін]. – К.:Просвіта. – 2008. – С. 413–431.

7. Графов Р.П. Использование нечеткой логики для поиска троянских программных продуктов в вычислительных системах / Р.П. Графов, О.С. Савенко, С.М. Лисенко // Вісник Чернівецького національного університету. – 2009. – № 6. – С. 85–91.

8. Савенко О.С. Алгоритми пошуку троянських програм в персональних комп'ютерах / О.С. Савенко, С.М. Лисенко // Радіоелектронні і комп'ютерні системи. – 2009. – № 5. – С. 120–126.

9. Савенко О. Розробка процесу виявлення троянських програм на основі використання штучних імунних систем / Олег Савенко, Сергій Лисенко // Вісник Хмельницького національного університету. – 2008. – № 5. – С. 183–188.

Рецензент: к.т.н. Косенков В.Д.
Надійшла 13.2.2012 р.

УДК 004.891.3: 004.3

Т.О. ГОВОРУЩЕНКО, А.В. БАЧИНСЬКИЙ
Хмельницький національний університет

ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ МЕТРИК СКЛАДНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Авторами статті обрано метрики складності ПЗ етапу проектування та досліджено граничні значення метрик складності, а також проведено оцінювання ефективності кожної метрики складності ПЗ етапу проектування з точки зору оцінювання значущості її інформації за адитивним критерієм ефективності.

The authors chose the design stage software complexity metrics, investigated the limits of complexity metrics, conducted the evaluating of the effectiveness of each design stage software complexity metric in terms of its information relevance assessment for additive criterion of efficiency.

Ключові слова: програмне забезпечення (ПЗ), етап проектування ПЗ, метрики складності ПЗ етапу проектування, граничні значення метрик складності ПЗ, значущість метрик складності ПЗ, кількість інформації метрик складності ПЗ, ефективність метрик складності ПЗ, адитивний критерій ефективності.

Вступ

Сучасна індустрія програмного забезпечення характеризується високою конкуренцією. Для успішної роботи на цьому ринку софтверна компанія повинна розробляти, впроваджувати та супроводжувати ПЗ швидко, вкладаючись в термін, та із задовільною складністю. Ряд софтверних компаній вкладають значні кошти в модернізацію процесів розроблення ПЗ з метою підвищення його якості.

За особливостями і властивостями життєвого циклу програм їх доцільно ділити на ряд класів і категорій, з яких найбільш розрізняються два великих класи – малі й великі, що визначаються кількістю рядків програмного коду [1].

До класу програм малого розміру належать програми, які створюються одним або декількома (3–5)

програмістами та мають відносно невелику кількість рядків. Малі програми створюються переважно для одержання конкретних результатів та аналізу відносно простих процесів. Вони не призначені для масового поширення, не мають конкретного незалежного замовника-споживача, не обмежуються замовником певною вартістю, трудомісткістю, вимогами заданої якості та термінами їх створення та не підлягають незалежному тестуванню, гарантуванню якості та сертифікації. До малих програм можна віднести програми, призначені для проміжних розрахунків, для потреб програміста і т.і.

Клас великих програм складають масштабні проекти для складних систем управління та обробки інформації, які оформляються у вигляді програмних продуктів з гарантованою якістю. Великі програми мають великий розмір та високу вартість. Від замовника великої програми або програмної системи розробники повинні одержати вимоги до характеристик та функційності продукту. Від розробників проектів вимагаються гарантії високої якості, надійності функціонування та безпеки застосування компонентів і програмних систем. До великих програм відноситься системне програмне забезпечення, а також деяке прикладне програмне забезпечення, наприклад, офісні пакети, графічні редактори і т.і. Великі програми називають ще програмними системами (ПС) або інформаційними системами (ІС).

Програмна складність характеризується довжиною програми або обсягом пам'яті ЕОМ, необхідної для розміщення ПЗ [2].

Структурна складність програм визначається кількістю взаємодіючих компонент, кількістю зв'язків між компонентами та складністю їх взаємодії [2].

Складність деякого міжмодульного зв'язку в процесі проектування можна характеризувати ймовірністю помилки при її формалізації та ступенем впливу цієї помилки на наступне функціонування модулів [2].

Складність програмних модулів характеризується конструктивною складністю створення оформленої компоненти програми і оцінюється з позиції складності внутрішньої структури та перетворення змінних в кожному модулі, а також інтегрально за деякими зовнішніми статичними характеристиками модулів [2].

Складність програми для систем реального часу переважно визначається допустимим часом відгуку, а для інформаційних систем – кількістю типів оброблюваних змінних [2].

Одним з основних засобів аналізу та оцінювання складності ПЗ є метричний аналіз. Метрика визначається як міра ступеня володіння властивістю, яка має числове значення [3]. Метрика ПЗ – це міра, яка дозволяє одержати числове значення деякої властивості ПЗ або його специфікацій. Сучасна програмна індустрія накопичила велику кількість метрик, які оцінюють окремі виробничі та експлуатаційні властивості ПЗ. Однак прагнення їх універсальності, неврахування типу та області застосування розроблюваного ПЗ, ігнорування етапів життєвого циклу ПЗ та необґрунтоване їх використання в процедурах прийняття виробничих рішень істотно підірвало довіру розробників та користувачів ПЗ до метрик. Ці обставини вимагають ретельного відбору метрик для певного типу та області застосування розроблюваного ПЗ, врахування їх обмежень на різних етапах життєвого циклу ПЗ, встановлення порядку їх сумісного використання, накопичення та інтеграції різнорідної метричної інформації для прийняття своєчасних виробничих рішень.

Отже, методи оцінки складності ПЗ, особливо на етапі проектування, на сьогодні є суб'єктивно залежними, для знаходження значень метрик використовуються експертні вагові коефіцієнти; порівняння значень метрик поточних проектів з попередніми (постає проблема, що робити, якщо проект принципово новий); немає загальноприйнятої номенклатури метрик; відсутні точні значення метрик, з якими можна було б порівняти поточні одержані значення; при виборі проекту враховується не стільки інформація про складність проекту та майбутнього ПЗ, скільки інформація про тривалість, вартість, технології програмування та репутацію фірми-розробника.

1. Метрики складності програмного забезпечення

У роботах [3–5] були визначені метрики складності етапу проектування ПЗ з точними та прогнозованими значеннями (таблиця 1).

Таблиця 1

Метрики складності етапу проектування ПЗ з точними та прогнозованими значеннями

№ п/п	Метрики складності етапу проектування з точними значеннями	Метрики складності етапу проектування з прогнозованими значеннями
1	Метрика Чепіна	Очікувана LOC-оцінка
2	Метрика Джилба (абсолютна модульна складність програми)	Метрика Холстеда (складність програми)
3	Метрика Мак-Клура (загальна складність програмної системи)	Метрика Маккейба (цикломатична складність ПЗ в цілому)
4	Метрика Кафура (інформаційна складність модуля)	Метрика Джилба (відносна логічна складність програми)
5		Прогнозована кількість операторів програми
6		Прогнозована оцінка складності інтерфейсів ПЗ

Розглянемо методику розрахунку вищенаведених метрик складності ПЗ етапу проектування.

Метрика Чепіна аналізує характер використання змінних зі списку введення, тобто оброблювану інформацію. Множина змінних з врахуванням їх значущості обчислюється як:

$$Q = P + 2M + 3C + 0{,}5T, \tag{1}$$

де P – змінні для розрахунків і виведення;

M – модифіковані або створені в програмі змінні;

C – керуючі змінні;

T – не використовувані в програмі ("паразитні") змінні. Метрика Чепіна дає оцінку інформаційної міцності модуля.

Метрика Джилба (складова метрики – модульна складність програми) – на етапі проектування можна підрахувати кількість міжмодульних зв'язків N_{zv} (абсолютна модульна складність) та відношення кількості міжмодульних зв'язків до кількості модулів L_{mod}: $f = \dfrac{N_{zv}^4}{L_{\text{mod}}}$ – відносна модульна складність. Чим вище значення відносної модульної складності проекту, тим вищий ступінь має зчеплення модулів проекту, а, отже, й складнішим є програмний проект.

За іншою методикою кількість умовних операторів (L_{IF}) та операторів циклу (L_{LOOP}) складає абсолютну логічну складність програми

$$CL = L_{IF} + L_{LOOP}, \tag{2}$$

а відношення абсолютної логічної складності програми до загальної кількості операторів L складає відносну логічну складність програми

$$cl = \frac{CL}{L}. \tag{3}$$

Метрика Мак-Клура – метрика, спрямована на оцінювання архітектури системи; міра складності, заснована на кількості можливих шляхів виконання програми, кількості керуючих конструкцій і змінних. Обчислюється в 3 етапи:

1) для кожної керуючої змінної i обчислюється значення її функції складності

$$C(i) = (D(i) * J(i))/n, \tag{4}$$

де $D(i)$ – величина, яка вимірює сферу дії змінної (для локальної змінної $D(i) = 0$, а для глобальної змінної – $D(i) = 1$);

$J(i)$ – міра складності взаємодії модулів через цю змінну (скільки разів модулі взаємодіяли з використанням цієї змінної);

n – кількість модулів;

2) для всіх модулів визначаються функції складності

$$M(P) = fp * X(P) + gp * Y(P), \tag{5}$$

де fp, gp – відповідно кількість модулів, безпосередньо передуючих і безпосередньо слідуючих за модулем P;

$X(P)$ – складність звертання до модуля P (скільки разів відбувається звертання до модуля P);

$Y(P)$ – складність управління викликом з модулю P інших модулів (скільки разів модуль P викликає інші модулі);

3) загальна складність MP програмної системи:

$$MP = \sum_P M(P). \tag{6}$$

Дана метрика орієнтована на добре структуроване програмне забезпечення, для якого можна побудувати схему розбиття ПЗ на модулі. В кожному модулі передбачена одна точка входу і одна точка виходу, модуль виконує одну функцію, виклик модуля здійснюється відповідно до ієрархічної системи керування. Метрика Мак-Клура дозволяє обрати схему розбиття з меншою складністю ще до написання програми.

Метрика Кафура – метрика, заснована на врахуванні потоків даних. Для розрахунку цієї метрики вводяться поняття локального і глобального потоків даних. Якщо модуль А викликає модуль В, то це прямий локальний потік з А в В; якщо модуль В викликає модуль А, і модуль А повертає значення в модуль В – це непрямий локальний потік з А в В; якщо модуль С викликає модулі А і В та передає результат з модуля А в модуль В – це локальний потік з А в В. Глобальний потік з модуля А в модуль В крізь глобальну структуру даних D існує, якщо модуль А поміщає інформацію в структуру D, а модуль В використовує інформацію зі структури D. Тоді:

1) інформаційна складність процедури обчислюється за формулою:

$$I = length \times (fan\_in \times fan\_out)^2, \tag{7}$$

де $length$ – складність тексту процедури (вимірюється однією з метрик складності);

fan_in – кількість локальних потоків всередину процедури плюс кількість структур даних;

fan_out – кількість локальних потоків з процедури плюс кількість структур даних, які оновлюються процедурою;

2) інформаційна складність модуля відносно деякої структури даних обчислюється за формулою:

$$I = W \times R + W \times WrRd + WrRd \times R + WrRd \times (WrRd - 1), \quad (8)$$

де W – кількість процедур, які оновлюють структуру даних, R – кількість процедур, які читають інформацію зі структури даних, $WrRd$ – кількість процедур, які читають і оновлюють структуру даних.

Очікувана LOC-оцінка (експертна) – за кожною функцією експерти надають краще, гірше та імовірне значення, тоді очікувана LOC-оцінка (LOC – кількість рядків вихідного коду програми) обчислюється за формулою:

$$LOC_{оч_i} = (LOC_{кращ_i} + LOC_{гірш_i} + 4 \times LOC_{імов_i}) / 6. \quad (9)$$

Показник LOC залежить від мови програмування. Це оціночна і необов'язкова метрика. Вона може призвести до оптимізації кількості рядків коду, а не проекту в цілому. Дозволяє спрогнозувати трудовитрати, час і вартість розробки, а також необхідну кількість програмістів для виконання проекту. Є вільно поширювані інструменти очікуваної LOC-оцінки. LOC-оцінка впливає на оцінку величини змін обсягу коду в часі.

Метрики Холстеда:

1) міра довжини модуля

$$N = n_1 \log_2(n_1) + n_2 \log_2(n_2), \quad (10)$$

де n_1 – кількість різних операторів, n_2 – кількість різних операндів;

2) обсяг модуля як кількість символів для запису всіх операторів і операндів тексту програми

$$(V = N \times \log_2(n_1 + n_2)). \quad (11)$$

Метрики Холстеда обчислюються на основі аналізу кількості рядків та синтаксичних елементів вихідного коду програми.

Основу метрик Холстеда складають 4 вимірювані характеристики програми: $NUOprtr$ – кількість унікальних операторів програми включно з іменами підпрограм (словник операторів), $NUOprnd$ – кількість унікальних операндів програми (словник операндів), $NOprtr$ – загальна кількість операторів в програмі, $NOprnd$ – загальна кількість операндів програми.

На основі цих характеристик обчислюються наступні оцінки:

1) словник програми:

$$HPVoc = NUOprtr + NUOprnd; \quad (12)$$

2) довжина програми:

$$HPLen = NOprtr + NOprnd; \quad (13)$$

впливає на оцінку величини змін обсягу коду в часі;

3) обсяг програми:

$$HPVol = HPLen \times \log_2 HPVoc; \quad (14)$$

впливає на оцінку величини змін обсягу коду в часі;

4) складність програми:

$$HDiff = (NUOprtr / 2) \times (NOprnd / NUOprnd); \quad (15)$$

5) оцінка зусиль програміста при розробці програми:

$$HEff = HDiff \times HPVol \quad (16)$$

вказує, наскільки ефективна праця розробника. Використовується для визначення складності реалізації певної вимоги, функції, модуля, частини проекту. Впливає на точність прогнозів оцінки трудомісткості та на розуміння того, наскільки інтелектуально-витратною для розробника буде та чи інша функція.

Метрика Маккейба – цикломатична складність. Один з найбільш розповсюджених показників оцінки складності програмних проектів. Цикломатичне число Маккейба показує необхідну кількість проходів для покриття всіх контурів сильнозв'язаного графу керування програми, тобто кількість тестових прогонів програми, необхідних для проведення вичерпного тестування. Дозволяє провести оцінку трудомісткості реалізації окремих елементів програмного проекту, оцінку трудомісткості тестування програмного проекту, скоригувати загальні показники оцінки тривалості та вартості проекту, оцінити зв'язані ризики і прийняти необхідні управлінські рішення.

Метрика Маккейба обчислюється для оцінки складності ПС, виходячи з топології внутрішніх зв'язків. Обчислення метрики проводяться на основі графу керування програми (орграф, в якому обчислювальні оператори або вирази представляються вершинами, а передачі керування між вершинами представляються дугами). Спрощена оцінка цикломатичної складності

$$V(G) = E - N + 2, \quad (17)$$

де E – кількість дуг, а N – кількість вершин в керуючому графі ПЗ (логічні оператори не враховуються).

Автоматизоване обчислення метрики Маккейба зводиться до підрахунку кількості логічних операторів та можливої кількості шляхів виконання програми.

Метрика Маккейба може бути обчислена для модуля, методу та інших структурних одиниць програмного проекту. Вона впливає на оцінку змін складності в часі.

Прогнозована кількість операторів програми:

$$N_{прогн} = NF \times N_{од}, \tag{18}$$

де NF – кількість функцій або вимог в специфікації програми, $N_{од}$ – одиничне значення кількості операторів (середня кількість операторів які доводяться на одну функцію або вимогу).

Прогнозована оцінка складності інтерфейсів компонентів ПЗ:

$$C = \frac{NI}{NF \times NI_{од} \times K_{скл}}, \tag{19}$$

де NI – загальна кількість змінних, які передаються по інтерфейсах між компонентами програми, $NI_{од}$ – одиничне значення кількості змінних, які передаються по інтерфейсах між компонентами (середня кількість змінних, які передаються по інтерфейсах, що доводяться на одну функцію або вимогу); обирають на основі аналізу статистичних даних; $K_{скл}$ – коефіцієнт складності розроблюваної програми, який враховує ріст одиничної складності програми (складності, що доводиться на одну функцію або вимогу специфікації) в порівнянні з середнім показником складності; його слід обирати з врахуванням статистичних даних та характеристик розробленої ПЗ.

2. Розрахунок граничних значень метрик складності програмного забезпечення етапу проектування

Для розрахунку максимальних значень метрик складності ПЗ слід ввести певні обмеження на проекти та ПЗ, метрики складності яких аналізуватимуться. Введемо наступні обмеження:

1) у кожному модулі ПЗ є максимум 100 змінних для розрахунків і виведення, 100 модифікованих або створених в модулі змінних, 100 керуючих змінних, 100 не використовуваних в модулі ("паразитних") змінних – всього 400 змінних;

2) ПЗ має не більше 50 модулів;

3) кожен модуль ПЗ пов'язаний з іншими 49 модулями одним зв'язком;

4) в кожному модулі є не більше 50 процедур, які оновлюють структуру даних, не більше 50 процедур, які читають інформацію зі структури даних, та не більше 50 процедур, які читають і оновлюють структуру даних;

5) ПЗ містить не більше 50000 рядків вихідного коду;

6) кількість унікальних операторів програми включно з іменами підпрограм (словник операторів) не перевищує 25000 (кожен другий рядок коду – це унікальний оператор), загальна кількість операндів програми не перевищує 50000 (в кожному рядку коду є один операнд), кількість унікальних операндів програми (словник операндів) не перевищує 400;

7) кожен рядок програми – це логічний оператор або оператор циклу;

8) кожна змінна модуля передається по його інтерфейсу.

Тоді *метрика Чепіна* для одного модуля матиме максимальне значення $Q_m = 100 + 2 \cdot 100 + 3 \cdot 100 + 0{,}5 \cdot 100 = 650$ згідно формули (1). Максимальне значення метрики Чепіна для всього ПЗ становить $Q = 50 \cdot 650 = 32500$.

Максимальна кількість міжмодульних зв'язків (абсолютна модульна складність програми) за метрикою Джилба (формула (2)) становитиме $50 * 49 = 2450$. Максимальна абсолютна логічна складність програми становить 50000, а максимальна *відносна логічна складність програми* в такому разі становить за формулою (3) $CL = \frac{50000}{50000} = 1.$

Враховуючи вищевикладені припущення щодо кількості модулів та кількості міжмодульних зв'язків, функція складності кожного з модулів для *метрики Мак-Клура* матиме значення за формулою (4.2):

$$M(P_1) = 0 \cdot 49 + 49 \cdot 49 = 2401$$
$$M(P_2) = 1 \cdot 49 + 48 \cdot 49 = 2401$$
$$M(P_3) = 2 \cdot 49 + 47 \cdot 49 = 2401$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$M(P_{50}) = 49 \cdot 49 + 0 \cdot 49 = 2401$$

Тоді максимальне значення загальної складності програмної системи (метрика Мак-Клура) становить за формулою (4.3): $M(P) = 50 * 2401 = 120050$.

За *метрикою Кафура* одержимо максимальне значення інформаційної складності одного модуля за

формулою (5.2) $I_m = 50 \cdot 50 + 50 \cdot 50 + 50 \cdot 50 + 50 \cdot (50-1) = 9950$. Максимальне значення інформаційної складності проекту становить $I = 50 * 9950 = 497500$.

Максимальне значення *очікуваної LOC-оцінки* становить 50000 рядків коду, враховуючи вищевикладені припущення.

Максимальна складність програми за метрикою Холстеда згідно формули (7.6) становить $HDiff = \frac{25000}{2} \cdot \frac{50000}{400} = 1562500$.

Максимальна *цикломатична складність Маккейба* за формулою (8): $V(G) = 2450 - 50 + 2 = 2402$.

Якщо кожен рядок коду – це один оператор, тоді максимальна *прогнозована кількість операторів програми* становить 50000.

Враховуючи припущення щодо кількості змінних в модулі, обчислимо *максимальну прогнозовану оцінку складності інтерфейсів ПЗ* за формулою (10): $C = \frac{20000}{50 \cdot 400 \cdot K_{скл}} = \frac{1}{K_{скл}}$. Оскільки $K_{скл} = 1 + \sum_{i=1}^{n} K_i$, де K_i – приріст складності розроблюваної програми за *i*-ю характеристикою, n – кількість врахованих характеристик, то максимальна прогнозована оцінка складності інтерфейсів ПЗ становить 1.

3. Оцінювання ефективності використання метрик складності ПЗ на етапі проектування

Розрахунок максимальних значень метрик складності ПЗ етапу проектування показав, що існує великий розкид діапазонів значень вхідних даних – значення вхідних векторів розрізняються в десятки, сотні, тисячі та навіть мільйони разів, що ускладнює аналіз метрик складності ПЗ етапу проектування. Опрацювання метрик в природньому вигляді призводитиме до того, що визначальними при аналізі складності проектів та ПЗ є метрики з великими діапазонами значень (метрики Холстеда, Кафура, Мак-Клура), метрики з середніми діапазонами значень (метрика Чепіна, прогнозована LOC-оцінка, прогнозована кількість операторів програми) менше впливатимуть на результати оцінювання рівня складності проектів та ПЗ, а метрики з малими діапазонами значень (метрики Джилба, Маккейба, прогнозована оцінка складності інтерфейсів) взагалі не впливатимуть на оцінку рівня складності проекту та ПЗ, тобто відбувається втрата частини значущої інформації про складність проекту та ПЗ.

Проведемо оцінку ефективності метрик складності ПЗ з точки зору оцінювання їх інформації.

Критеріями оцінювання інформації є [6]:

1) релевантність – наявність зв'язку з проблемою (відповідність інтересам та проблемі) та здатність інформації внести ясність у процес розуміння проблеми. Для оцінювання релевантності інформації слід співставити її з інформаційними потребами та відповісти на питання, чи зможе ця інформація допомогти в найближчому майбутньому. Ознаки релевантності: наявність явної вказівки на сферу інтересів через наявність ключових слів; наявність непрямої, змістовної вказівки;

2) достовірність – на скільки представлений опис відповідає дійсності: або можна вірити інформації, або потрібне додаткове дослідження, або не можна довіряти в принципі. Достовірність перевіряється за наступними параметрами: наявність підтвердження з інших джерел; сумісність з іншою інформацією; знання джерела та його мотивів; авторитет або тривала позитивна історія роботи з джерелом; властивості каналу передачі інформації. Очевидно, що визначальне місце тут посідає знання про джерело інформації, тому важливо вести роботу по вивченню джерел, постійному збиранню інформації про них;

3) значущість – розуміння інформації, повнота висвітлення предмету інтересу, своєчасність інформації та її достатність для прийняття рішень.

Щодо *релевантності* метрик складності ПЗ проблемі оцінювання складності проектів та ПЗ, то висновок однозначний – інформація релевантна, причому рівень релевантності найвищий (дорівнює 1), оскільки саме метрики складності дають уявлення про рівень складності проекту і ПЗ.

Достовірність інформації про метрики складності ПЗ максимальна (дорівнює 1), оскільки використовуються лише класичні метрики складності, описані в різноманітних галузевих публікаціях.

Значущість інформації може визначатись кількістю інформації [7] – як зменшення ентропії джерела інформації:

$$I = \log_2 P(A), \tag{20}$$

де $P(A)$ – ймовірність настання однієї з подій після одержання інформації. Є ще один спосіб розрахунку кількості інформації:

$$I = H(апріорна) - H(апостеріорна), \tag{21}$$

де $H(апріорна)$ – ентропія системи (показник незнання системи) до експерименту, $H(апостеріорна)$ – ентропія системи після експерименту, тоді, якщо $I > 0$, то в результаті експерименту одержано нову інформацію, показник незнання системи зменшився.

Оскільки релевантність і достовірність метрик складності ПЗ є сталими величинами, то основним

критерієм оцінювання інформації, який впливатиме на оцінку ефективності метрик складності ПЗ, є саме значущість інформації.

Існують різні підходи до формування критеріїв ефективності [8]: 1) формування одного підсумкового критеріального показника за адитивним або мультиплікативним критерієм; 2) відношення частини параметрів ефекту, які потрібно збільшити, до частини параметрів ефекту, які потрібно зменшити; 3) максимізація або мінімізація одного з параметрів ефекту та накладання обмежень на інші параметри ефекту.

Для визначення ефективності метрик складності ПЗ етапу проектування використаємо перший підхід – формування підсумкового критеріального показника для кожної метрики за адитивним критерієм.

Адитивний критерій A формується шляхом ділення на число показників ефекту n суми добутків часткових показників ефекту l_i на коефіцієнти значущості i-го показника g_i:

$$A = (1/n) \cdot \sum_{i=1}^{n} (l_i \cdot g_i). \qquad (22)$$

Оскільки ефективність кожної метрики складності ПЗ етапу проектування – це значущість інформації метрики, то вона залежить від 4-х показників ефекту ($n = 4$): розуміння інформації, повнота висвітлення предмету інтересу, своєчасність інформації та її достатність для прийняття рішень, кількість інформації. Кожен показник ефекту розглядатимемо для спрощення розрахунків як значення з інтервалу [0, 1], де 0 – мінімальне значення показника, 1 – максимальне значення. Значущості показників ефекту для кожної метрики рівні, причому $\sum_{i=1}^{n} g_i = 1$, отже, $g_i = \frac{1}{4} = 0{,}25$. Потрібно тепер розрахувати 4 часткових показники ефекту l_i для кожної метрики складності ПЗ етапу проектування і відповідно ефективність кожної метрики за адитивним критерієм.

Для показника "розуміння інформації" приймемо максимальне значення для кожної з метрик, оскільки вважатимемо, що метрики розраховані вірно, з повним розумінням. Показник "повнота висвітлення предмету інтересу" може досягати максимального значення в залежності від повноти висвітлення складності ПЗ на етапі проектування. Інформаційні потоки етапу проектування ПЗ [9]: вимоги до ПЗ, представлені інформаційною, функційною та поведінковою моделями аналізу. Інформаційна модель описує інформацію, яку повинне обробляти ПЗ на думку замовника. Функційна модель визначає перелік функцій обробки інформації та перелік модулів програмної системи. Поведінкова модель фіксує бажану динаміку системи (режими її роботи). На виході етапу проектування – розробка даних, розробка архітектури та процедурна розробка ПЗ. Отже, повнота висвітлення складності ПЗ на етапі проектування залежить від повноти висвітлення метрикою складності: оброблюваної інформації, функцій обробки інформації та переліку модулів, режимів роботи ПЗ. Показник "своєчасність інформації та її достатність для прийняття рішень" може досягати максимального значення в залежності від достатності інформації про оброблювану інформацію, функції обробки інформації та перелік модулів, режими роботи ПЗ. Щодо кількості інформації, то вважатимемо, що після опрацювання всіх метрик складності показник незнання складності проекту та ПЗ стає рівним 0, тому кількість інформації I, яку містять всі метрики складності, згідно формули (12) дорівнює 1. Вважатимемо також, що показник незнання складності ПЗ зменшується в рівному ступені після опрацювання кожної метрики, тому кількість інформації кожної метрики складності $I_м = \frac{1}{9} = 0{,}111$.

Метрика Чепіна. Розуміння інформації $l_{1_1} = 1$, повнота висвітлення предмету інтересу $l_{2_1} = 0{,}333$ (аналізується лише оброблювана інформація), своєчасність інформації та її достатність для прийняття рішень $l_{3_1} = 0{,}333$, кількість інформації $l_{4_1} = 0{,}111$, тоді за формулою (13):

$$A_1 = (1/4) \cdot \sum_{i=1}^{4} (l_{i_1} \cdot 0{,}25) = 0{,}25 \cdot (1 \cdot 0{,}25 + 0{,}333 \cdot 0{,}25 + 0{,}333 \cdot 0{,}25 + 0{,}111 \cdot 0{,}25) = 0{,}111$$

Метрика Джилба (абсолютна). Розуміння інформації $l_{1_2} = 1$, повнота висвітлення предмету інтересу $l_{2_2} = 0{,}333$ (до уваги приймається кількість модулів та функції обробки інформації), своєчасність інформації та її достатність для прийняття рішень $l_{3_2} = 0{,}333$, кількість інформації $l_{4_2} = 0{,}111$, тоді за формулою (13): $A_2 = 0{,}111$

Метрика Джилба (відносна). Розуміння інформації $l_{1_3} = 1$, повнота висвітлення предмету інтересу $l_{2_3} = 0{,}333$ (до уваги приймається кількість модулів та функції обробки інформації), своєчасність інформації та її достатність для прийняття рішень $l_{3_3} = 0{,}333$, кількість інформації $l_{4_3} = 0{,}111$, тоді за формулою (13): $A_3 = 0{,}111$

Метрика Мак-Клура. Розуміння інформації $l_{1_4}=1$, повнота висвітлення предмету інтересу $l_{2_4}=1$ (аналізується оброблювана інформація, функції оброблення інформації та кількість модулів, а також архітектура та режими роботи ПЗ), своєчасність інформації та її достатність для прийняття рішень $l_{3_4}=1$, кількість інформації $l_{4_4}=0{,}111$, тоді за формулою (13): $A_4=0{,}194$

Метрика Кафура. Розуміння інформації $l_{1_5}=1$, повнота висвітлення предмету інтересу $l_{2_5}=0{,}333$ (враховуються потоки даних), своєчасність інформації та її достатність для прийняття рішень $l_{3_5}=0{,}333$ (враховуються потоки даних), кількість інформації l_{4_5}, тоді за формулою (13): $A_5=0{,}111$

Очікувана LOC-оцінка. Розуміння інформації $l_{1_6}=1$, повнота висвітлення предмету інтересу $l_{2_6}=0{,}1665$ (метрика пов'язана з функціями оброблення інформації, але не враховує кількості модулів), своєчасність інформації та її достатність для прийняття рішень $l_{3_6}=0{,}1665$, кількість інформації $l_{4_6}=0{,}111$, тоді за формулою (13): $A_6=0{,}090$

Метрика Холстеда. Розуміння інформації $l_{1_7}=1$, повнота висвітлення предмету інтересу $l_{2_7}=0{,}1665$ (метрика пов'язана з функціями оброблення інформації, але не враховує кількості модулів), своєчасність інформації та її достатність для прийняття рішень $l_{3_7}=0{,}1665$, кількість інформації $l_{4_7}=0{,}111$, тоді за формулою (13): $A_7=0{,}090$

Метрика Маккейба. Розуміння інформації $l_{1_8}=1$, повнота висвітлення предмету інтересу $l_{2_8}=1$ (метрика враховує кількість модулів, функції оброблення інформації і кількість модулів та режими роботи ПЗ), своєчасність інформації та її достатність для прийняття рішень $l_{3_8}=1$, кількість інформації $l_{4_8}=0{,}111$, тоді за формулою (13): $A_8=0{,}194$

Прогнозована кількість операторів програми. Розуміння інформації $l_{1_9}=1$, повнота висвітлення предмету інтересу $l_{2_9}=0{,}1665$ (метрика пов'язана з функціями оброблення інформації, але не враховує кількості модулів), своєчасність інформації та її достатність для прийняття рішень $l_{3_9}=0{,}1665$, кількість інформації $l_{4_9}=0{,}111$, тоді за формулою. (13): $A_9=0{,}090$

Прогнозована оцінка складності інтерфейсів ПЗ. Розуміння інформації $l_{1_{10}}=1$, повнота висвітлення предмету інтересу $l_{2_{10}}=0{,}4995$ (враховується оброблювана інформація та функції її оброблення, але не враховується кількість модулів), своєчасність інформації та її достатність для прийняття рішень $l_{3_{10}}=0{,}4995$, кількість інформації $l_{4_{10}}=0{,}111$, тоді за формулою (13): $A_{10}=0{,}132$.

Отже, проведено оцінювання ефективності кожної з 10 метрик складності ПЗ етапу проектування з точки зору оцінювання значущості її інформації за адитивним критерієм ефективності.

Висновки

В ході дослідження обрано метрики складності ПЗ етапу проектування та досліджено граничні значення метрик складності, які розрізняються в десятки, сотні, тисячі та навіть мільйони разів, що призводить до визначальності впливу при аналізі складності проектів та ПЗ метрик з великими діапазонами значень (метрики Холстеда, Кафура, Мак-Клура), незначності впливу на результати оцінювання рівня складності проектів та ПЗ метрик з діапазонами значень середньої величини (метрики Чепіна, прогнозованої LOC-оцінки, прогнозованої кількості операторів програми) та відсутності впливу на оцінку рівня складності проекту та ПЗ метрик з малими діапазонами значень (метрики Джилба, Маккейба, прогнозованої оцінки складності інтерфейсів), тобто до втрати частини значущої інформації про складність проекту та ПЗ. Кількість втраченої інформації при неврахуванні трьох метрик з малими діапазонами значень становить $I=3\cdot 0{,}111=0{,}333$, тобто втрачається третина інформації про складність проекту та ПЗ.

Оцінки ефективності метрик складності ПЗ етапу проектування показали, що порівняно високу ефективність мають метрики Мак-Клура, Маккейба, оцінки складності інтерфейсів ПЗ; середню ефективність мають метрики Чепіна, Джилба (абсолютна, відносна), Кафура; низьку ефективність мають метрики очікуваної LOC-оцінки та прогнозованої кількості операторів програми. Отже, при оцінюванні складності ПЗ необхідно враховувати метрики з малими діапазонами значень, оскільки їх ефективність є значною.

Однак, необхідно попередньо опрацьовувати діапазони значень метрик. Перспективним є розроблення нового або модифікованого методу опрацювання інформації (масштабування, нормування,

факторного аналізу) що забезпечив би рівнозначне врахування метрик як з великими, так і з малими діапазонами.

Література

1. Липаев В.В. Программная инженерия. Методологические основы / Липаев В.В. – М. : ТЕИС, 2006. – 608 с

2. Ковалевская Е.В. Материалы к курсу "Метрология, качество и сертификация ПО" / Ковалевская Е.В. – М. : Московский государственный университет экономики, статистики и информатики, 2002. – 38 с.

3. Поморова О.В. Інтелектуальний метод оцінювання результатів проектування та прогнозування характеристик якості програмного забезпечення / О.В. Поморова, Т.О. Говорущенко // Радіоелектронні і комп'ютерні системи – Харків : НАУ "ХАІ", 2010 – № 6. – С. 211–218

4. Поморова О.В. Оцінювання результатів проектування та прогнозування характеристик якості програмного забезпечення / О.В. Поморова, Т.О. Говорущенко, О.С. Онищук // Вісник Хмельницького національного університету – Хмельницький : ХНУ, 2011 – № 2. С. 168–178

5. Поморова О.В. Аналіз методів та засобів оцінки якості програмних систем // Радіоелектронні і комп'ютерні системи / О.В. Поморова, Т.О. Говорущенко. – Харків : НАУ "ХАІ", 2009 – № 6. – С. 148–158

6. Нежданов И. Критерии оценки информации (важность, точность, значимость) [Електронний ресурс]. – Режим доступу : http://www.police-russia.ru/showthread.php?t=44683

7. Значимость информации: определение http://otvet.mail.ru/question/2703994

8. Белик Т.В. Принципы формирования критериев эффективности [Електронний ресурс] / Белик Т.В. – Режим доступу : http://lego.biuss.ru/paragraf/742

9. Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем : [учебник для ВУЗов] / Орлов С.А. – СПб. : Питер, 2004. – 527 с.

Рецензент: д.т.н. Поморова О.В.
Надійшла 9.2.2012 р.

УДК 681.31

І.Б. АЛБАНСЬКИЙ, О.І. ВОЛИНСЬКИЙ
Інститут мікропроцесорних систем НАН України

ДОСЛІДЖЕННЯ СИСТЕМНИХ ХАРАКТЕРИСТИК ЦИФРОВИХ ПРИСТРОЇВ МНОЖЕННЯ РЕАЛІЗОВАНИХ В РІЗНИХ ТЕОРЕТИКО-ЧИСЛОВИХ БАЗИСАХ

Викладені основи теоретико-числових базисів (ТЧБ), які використані для кодування, перетворення та цифрового опрацювання інформації. Класифіковані критерії оцінки системних характеристик цифрового пристрою множення (ЦПМ). Проведені дослідження системних характеристик цифрових пристроїв множення, які реалізовані в різних ТЧБ.

Expounding the virtues of theoretical and numerical bases (TNB) that used for encoding transform and digital processing of information. Classified criteria for evaluating system characteristics digital multiplication equipment (DME). Conducted studies n of system characteristics of digital devices multiplication, are implemented in different TNB.

Ключові слова: цифрові пристрої множення, теоретико-числові базиси, апаратна складність, часова складність.

Вступ

ЦПМ є важливими компонентами спецпроцесорів кореляційного опрацювання сигналів [1]. Особливо найбільш стандартними вимогами до системних характеристик таких компонентів є досягнення максимальної швидкодії операцій множення, зниження апаратної складності, високий рівень регулярності архітектур та глибокий ступінь розпаралелення обчислювальних операцій. В той же час при реалізації спецпроцесорів в різних ТЧБ крім найбільш відомого базису Радемахера суттєво розширює клас вимог та можливостей реалізації ЦПМ у сфері розширених застосувань [2].

Огляд літератури та постановка задачі дослідження

В роботах [3] викладені основи ТЧБ, які використовуються для побудови компонентів комп'ютерних систем табл.1.

З табл. 1 видно, що наступні ТЧБ: Унітарний та Хаара характеризуються надлишковістю представлених даних. Незважаючи на це при паралельних опрацюваннях даних у цифровій голографії, томографії, око-процесорах, побудові цифрових кореляторів [4] широко використовуються цифрові перемножувачі оскільки дозволяється глибоко розпаралелити обчислювальні процеси та спростити реалізацію пристроїв додавання, віднімання, множення та порівняння.

Більш ефективним по відношенню до унітарного базису Хаара при однаковому об'ємі кодової матриці по відношенню до унітарного базису характеризується більш ефективною у зв'язку з меншим числом активних елементів кодової матриці. Даний базис знаходить найширше застосування у системах