УДК 004.891.3: 004.3

О.В. ПОМОРОВА, Т.О. ГОВОРУЩЕНКО

Хмельницький національний університет, Україна

ІНТЕЛЕКТУАЛЬНИЙ МЕТОД ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ ПРОЕКТУВАННЯ ТА ПРОГНОЗУВАННЯ ХАРАКТЕРИСТИК ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті виконано аналіз метрик етапу проектування з точними та прогнозованими значеннями з метою підготовки їх до інтелектуального опрацювання, на основі якого запропоновано інтелектуальний метод оцінювання результатів проектування та прогнозування характеристик якості програмного забезпечення (ІМОП). Наведені оцінки результатів проектування характеризують рівень компанії, що розробляє програмне забезпечення, та серйозність ставлення компанії до даного замовлення, а також дають дані замовнику для вибору кращої софтверної компанії для розробки необхідного проекту та ПЗ.

***Ключові слова**: програмне забезпечення (ПЗ), етап проектування програмного забезпечення, метрики якості та складності програмного забезпечення, метрики етапу проектування ПЗ, інтелектуальний метод оцінювання результатів проектування та прогнозування характеристик якості ПЗ (ІМОП)*

Вступ

На сьогодні, програмне забезпечення (ПЗ) є визначальною складовою багатьох систем, серед яких системи критичного застосування та спеціалізовані системи різноманітного призначення. Для зазначених систем проблема наявності помилок в програмному забезпеченні та низька якість ПЗ загрожує катастрофами в певних галузях, які призводять до людських жертв та екологічних катаклізмів, або, щонайменше, до значних економічних та часових втрат. Проблема виявлення та усунення помилок загострюється по мірі збільшення складності задач та програм, які їх вирішують. Розвиток сучасних технологій розроблення ПЗ вимагає динамічного розвитку засобів оцінки характеристик якості ПЗ, причому з точки зору економічної та часової доцільності вже на етапі проектування, що дасть можливість обрати той рівень якості та ту вартість, які задовольнять замовника.

На етапі проектування програмного забезпечення (ПЗ) формується відповідь на питання "Яким чином програмна система буде реалізовувати висунуті до неї вимоги?" Інформаційні потоки етапу проектування: вимоги до ПЗ, представлені інформаційними потоками, які повинна обробляти ПС на думку замовника; переліком функцій обробки інформації та переліком модулів програмної системи; бажаною динамікою системи (режимами її роботи). Результатом етапу проектування є дані, архітектура та процедурна розробка ПЗ.

Згідно [1], метрика визначається як міра ступеня володіння властивістю, яка має числове значення. Взагалі, 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 необгрунтоване 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.

Незважаючи на численні дослідження програмних метрик , в галузі забезпечення якості ПЗ залишається *ряд невирішених питань* [2, 3]. За статистичними даними [4], лише 1.5% софтверних організацій намагаються оцінити якість процесів і готового продукту кількісно за допомогою метрик, і лише 0.5% софтверних організацій намагаються покращити роботу, керуючись кількісними критеріями якості ПЗ з метою випуску бездефектних продуктів. Технологія вимірювання якості ще не досягла зрілості, оскільки лише 0.5% софтверних організацій знаходяться на оптимізованому, зрілому рівні моделі CMM. Відсутні єдині стандарти на метрики, створено більше тисячі метрик [5], тому кожен постачальник "вимірювальної" системи пропонує власні спо-

© О.В. Поморова, Т.О. Говорущенко

собі оцінки якості і відповідно метрики. Існує також проблема складності інтерпретації величин метрик. Саме невирішеність цих питань не дозволяє підвищувати якість ПЗ та зменшувати його складність.

Приймаючи до уваги результати аналізу стандартів, моделей та метрик якості ПЗ, а також методів вимірювання показників якості ПЗ [2, 3], очевидно, що необхідно розробити методи та засоби оцінювання результатів проектування та прогнозування характеристик якості ПЗ. Перспективним напрямком досліджень є розроблення інтелектуальних систем, які: 1) обчислюватимуть розрахунковими та експертними методами точні або прогнозовані значення метрик програмного забезпечення вже на етапі проектування; 2) аналізуватимуть і опрацьовуватимуть результати метричних оцінок, на основі чого надаватимуть рекомендації, висновки і прогнози про розроблюване програмне забезпечення.

1. Аналіз метрик з точними значеннями на етапі проектування

В [2, 3] було проведено аналіз метрик якості програмного забезпечення з точки зору можливості їх застосування на етапі проектування ПЗ з одержанням точного або прогнозованого значення. В результаті такого аналізу було виділено ряд метрик, які вже на етапі проектування мають точне значення:

1) *метрика Чепіна* - аналізує характер використання змінних зі списку введення, тобто оброблювану інформацію. Множина змінних з врахуванням їх значущості обчислюється як: $Q = P + 2M + 3C + 0{,}5T$, 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;

2) *метрики зв'язності* (зв'язність (cohesion) - внутрішня характеристика програмного модуля) - залежать від типу модуля або проекту. На основі [6, 7, 8] опишемо наступні правила визначення рівня зв'язності модуля та сили зв'язності (СЗ):

- якщо модуль - одинична проблемно-орієнтована функція, то рівень зв'язності - функційний (частини модуля разом реалізують одну функцію) і СЗ=10;

- якщо дії всередині модуля зв'язані даними, і порядок дій всередині модуля важливий, то рівень зв'язності - інформаційний (вихідні дані однієї частини використовуються як вхідні дані в іншій частині модуля) і СЗ=9;

- якщо дії всередині модуля зв'язані даними, і порядок дій всередині модуля не має жодного значення, то рівень зв'язності - комунікативний (частини модуля зв'язані даними - працюють з однією структурою даних) і СЗ=7;

- якщо дії всередині модуля зв'язані потоком керування, і порядок дій всередині модуля важливий, то рівень зв'язності - процедурний (частини модуля зв'язані порядком виконуваних ним дій, які реалізують деякий сценарій поведінки) і СЗ=5;

- якщо дії всередині модуля зв'язані потоком керування, і порядок дій всередині модуля не має жодного значення, то рівень зв'язності - часовий (частини модуля не зв'язані, але необхідні в один і той же момент роботи системи) і СЗ=3; недоліком такого типу зв'язності є сильний взаємний зв'язок з іншими модулями, через що проект стає дуже чутливим до внесення змін;

- якщо дії всередині модуля жодним чином не зв'язані, але належать до однієї категорії, то рівень зв'язності - логічний (частини модуля об'єднані за принципом функційної подібності) і СЗ=1; недоліками логічної зв'язності є складне спряження та велика ймовірність внесення помилок при зміні спряження заради однієї з функцій;

- якщо дії всередині модуля жодним чином не зв'язані, а також не належать до однієї категорії, то рівень зв'язності - за співпаданням (в модулі відсутні явно виражені внутрішні зв'язки) і СЗ=0.

Можливі більш складні випадки, коли з модулем асоціюються декілька рівнів зв'язності. В такому випадку, враховуючи статистичну та експертну інформацію про тип та область застосування розроблюваного ПЗ, модулю присвоюють найсильніший рівень зв'язності або найслабший рівень зв'язності.

Чим вище зв'язність модуля, тим кращий результат проектування, тобто чим "чорніша" скринька проекту (захисна оболонка модуля), тим менше "важелів керування" на ній знаходиться і тим вона простіша. Такі типи зв'язності, як зв'язність за співпаданням, логічна зв'язність та часова зв'язність - результат невірного планування архітектури, а процедурний тип зв'язності - результат недбалого планування архітектури додатку;

3) *метрики зчеплення* (зчеплення (coupling) - зовнішня характеристика модуля, яку бажано і слід зменшувати) - міра взаємозалежності модулів за даними. Кількісно зчеплення вимірюється ступенем зчеплення (СЗч). На основі [6, 7, 8] виділимо наступні типи зчеплення:

- зчеплення за даними (СЗч=1) - один модуль викликає інший модуль, всі вхідні і вихідні параметри модуля, що викликається, - прості елементи даних;

- зчеплення за зразком (СЗч=3) - в якості вхідних і вихідних параметрів використовуються структури даних;

- зчеплення за керуванням (СЗч=4) - один модуль явно керує функціюванням іншого модуля за допомогою прапорців або перемикачів і надсилає йому керуючі дані;

- зчеплення за зовнішніми посиланнями (СЗч=5) - модулі посилаються на один і той же глобальний елемент даних;

- зчеплення за спільною областю (СЗч=7) - модулі поділяють одну й ту ж глобальну структуру даних;

- зчеплення за вмістом (СЗч=9) - один модуль прямо посилається на вміст іншого модуля не через його точку входу.

4) *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}: $f = \frac{N_{zv}^4}{L_{mod}}$ - відносна модульна складність [9].

Чим вище значення відносної модульної складності проекту, тим вищий ступінь має зчеплення модулів проекту, а, отже, й складнішим є програмний проект;

5) *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 [10]:

- для кожної керуючої змінної i обчислюється значення її функції складності $C(i) = (D(i) * J(i))/n$, 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;

- для всіх модулів визначаються функції складності $M(P) = fp * X(P) + gp * Y(P)$, 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);

- загальна складність MP програмної системи:

$$MP = \sum_P M(P). \qquad (1)$$

Дана метрика орієнтована на добре структуровані програмні системи, для яких мсожна побудувати схему розбиття ПС 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;

6) *метрика Кафура* - метрика, заснована на врахуванні потоків даних. Для розрахунку цієї метрики вводяться поняття локального і глобального потоків даних [10]. 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:

- інформаційна складність процедури обчислюється за формулою:

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

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;

- інформаційна складність модуля відносно деякої структури даних обчислюється за формулою:

$$I = W \times R + W \times WrRd + WrRd \times \times R + WrRd \times (WrRd - 1), \qquad (3)$$

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;

7) *метрика Зольновського, Сіммонса, Тейєра - складова метрики (структура, взаємодія, обсяг, дані)* - зважена сума індикаторів (структура, взаємодія, обсяг, дані) $\sum(a, b, g, n)$; гібридна метрика;

8) *метрика звертання до глобальних змінних - пара (модуль, глобальна змінна)* - $\text{пара}(p, r)$, де p - модуль, який має доступ до глобальної змінної r. В залежності від наявності реального звертання до змінної r формуються 2 типи пар (p, r) - фактичні і можливі. Характеристика Aup показує, скільки разів модуль дійсно одержить доступ до глобальної змінної, а характеристика Pup - скільки разів він міг би одержати такий доступ. Тоді наближену ймовірність посилання довільного модуля на довільну

глобальну змінну можна обчислити як відношення: $\mathrm{Rup} = \frac{\mathrm{Aup}}{\mathrm{Pup}}$. Чим вище така ймовірність, тим вище ймовірність "несанкціонованої" зміни певної глобальної змінної, що призводить до ускладнення модифікації програми;

9) *метрика Тая* - складність програми визначається вимірюванням інформаційного потоку даних. Досліджує вплив визначень даних та їх використання при наявності умовних та циклічних операторів, тобто вплив DU -пар (D - визначення даних, U - їх використання в умовних та циклічних операторах) на вибір виконуваної гілки програми ;

10) *метрика Вітворфа, Зулевського - складові метрики "міра складності потоку даних"* $W(P)$ та *"міра складності потоку керування"* $g(P)$;

11) *час модифікації моделей* - метрика процесу розробки ПЗ [11];

12) *кількість знайдених помилок при інспектуванні моделей та прототипів підсистем, модулів, функцій, вимог та густота помилок* (кількість помилок на одну підсистему, модуль, функцію, вимогу) - вказує на проблемну підсистему, модуль, функцію, вимогу. Це метрика процесу розробки ПЗ [11].

2. Аналіз метрик з прогнозованими значеннями на етапі проектування

В [2, 3] було виділено метрики етапу проектування ПЗ з прогнозованим значенням:

1) *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:

$$\mathrm{LOC}_{\text{î÷i}} = (\mathrm{LOC}_{\text{êðàù}} + \mathrm{LOC}_{\text{ãàðð}} + 4 \times \mathrm{LOC}_{\text{çì îæ}}) / 6. \quad (4)$$

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 [12]. LOC-оцінка впливає на оцінку величини змін обсягу коду в часі;

2) *метрики Холстеда:*

- міра довжини модуля

$$\mathrm{N} = \mathrm{n}_1 \log_2(\mathrm{n}_1) + \mathrm{n}_2 \log_2(\mathrm{n}_2), \quad (5)$$

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

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

$$\mathrm{V} = \mathrm{N} \times \log_2(\mathrm{n}_1 + \mathrm{n}_2). \quad (6)$$

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

Основі 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 обчсилюються наступні оцінки:

- словник програми

$$\mathrm{HPVoc} = \mathrm{NUOprtr} + \mathrm{NUOprnd}; \quad (7)$$

- довжина програми

$$\mathrm{HPLen} = \mathrm{NOprtr} + \mathrm{NOprnd}; \quad (8)$$

впливає на оцінку величини змін обсягу коду в часі

- обсяг програми

$$\mathrm{HPVol} = \mathrm{HPLen} \times \log_2 \mathrm{HPVoc}; \quad (9)$$

впливає на оцінку величини змін обсягу коду в часі

- складність програми

$$\mathrm{HDiff} = (\mathrm{NUOprtr}/2) \times (\mathrm{NOprnd}/\mathrm{NUOprnd}); \quad (10)$$

- оцінка зусиль програміста при розробці програми $\mathrm{HEff} = \mathrm{HDiff} \times \mathrm{HPVol}$ - вказує, наскільки ефективна праця розробника. Вкористовується 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.

3) *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 (оргграф, в якому обчислювальні оператори або вирази представляються вершинами, а передачі керування між вершинами

представляються дугами). Спрощена оцінка цикломатичної складності $V(G) = E - N + 2$, 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;

4) *метрика Хансена - пара (цикломатична складність Маккейба, кількість операторів)* - $(V(G), N)$;

5) *метрика Кокола* - комплексна метрика, яка, як правило, враховує значення метрик Холстеда, Маккейба і LOC:

$$H\_M = \frac{M + R_1 \times M(M_1) + ... + R_n \times M(M_n)}{1 + R_1 + ... + R_n}, \quad (11)$$

де M - базова метрика (метрика Холстеда), M_i - інші метрики (з точки зору автора, до таких метрик належать метрика Маккейба і LOC-оцінка), $M(M_i)$ - функції, обчислені за допомогою регресійного аналізу, R_i - коефіцієнти, обчислені за допомогою регресійного аналізу;

6) *метрика Джилба (складова - логічна складність програми)* - кількість умовних операторів (L_{IF}) та операторів циклу (L_{LOOP}) складає абсолютну логічну складність програми $CL = L_{IF} + L_{LOOP}$, а відношення абсолютної логічної складності програми до загальної кількості операторів L складає відносну логічну складність програми $cl = \frac{CL}{L}$;

7) *загальний час розробки і окремий час для кожного етапу* - метрика процесу розробки ПЗ [11];

8) *час виконання робіт в процесі* - метрика процесу розробки ПЗ [11];

9) *прогнозована кількість операторів програми* - $N_{прогн} = NF \times N_{од}$, 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);

10) *прогнозована оцінка складності інтерфейсів компонентів ПС:*

$$C = \frac{NI}{NF \times NI_{од} \times K_{скл}}, \quad (12)$$

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 ПС;

11) *очікувана вартість розробки кожної функції*: $ВАРТІСТЬ_i = LOC_{оч_i} \times ВАРТІСТЬ_{рядка_i}$; вартість рядка є константою і не змінюється від реалізації до реалізації;

12) *прогнозована вартість перевірки якості* - метрика процесу розробки ПЗ [11];

13) *прогнозована вартість процесу розробки* - метрика процесу розробки ПЗ [11];

14) *прогнозована продуктивність розробки кожної функції* (на основі продуктивності аналогічних функцій в аналогічних програмних продуктах):

$$ПРОДУКТ_i = ПРОДУКТ_{ан_i} \times (LOC_{ан_i} / LOC_{оч_i})$$

15) *прогнозовані витрати на розробку кожної функції*: $ВИТРАТИ_i = (LOC_{оч_i} / ПРОДУКТ_i)$;

16) *прогнозований функційний розмір FP* - вимірює суть можливостей майбутньої програми. Для обчислення функційного розміру: ідентифікуються очікувані від програмного додатку функції за критеріями International Function Point Users Group (IFPUG) [13]. Метод визначення функційного розміру [14] опишемо покроково наступним чином:

- виділити функції додатку;

- для кожної потенційної виділеної функції слід порахувати кількість зовнішніх входів EI, які по-різному впливають на виконувану функцію; кількість зовнішніх виходів EO, для істотно різних алгоритмів і нетривіальної функційності; кількість зовнішніх запитів EIN; кількість внутрішніх логічних файлів або унікальних логічних груп користувацьких даних ILF; кількість зовнішніх логічних файлів або унікальних логічних груп користувацьких даних ELF;

- кожен з визначених на попередньому кроці факторів множиться на коефіцієнт, який визначається складністю даного фактору в програмному проекті [14] (EI - 3 або 4 або 6 (від простого до складного), EO - 4 або 5 або 7, EIN - 3 або 4 або 6, ILF - 7 або 10 або 15, ELF - 5 або 7 або 10). Ці добутки додаються за кожною функцією;

- визначити ваги для 14 загальних характеристик проекту (від 0 до 5) [14, 15]; ваги вказуються в

формі діапазонів, що відображає невпевненість відносно функцій; призначення ваг вимагає певного досвіду у використанні методу функційного розміру;

- обчислити уточнений функційний розмір за формулою:

$$\text{Уточн.функц.розмір} = \text{Наближений\_функц\_розмір} \times [0.65 + 0.01 \times \times (\text{Сума\_загальних\_характеристик}] \quad (13)$$

Якщо до проекту не висувається жодних спеціальних вимог (всі загальні характеристики дорівнюють 0), то неуточнений функційний розмір слід зменшити на 35%, інакше слід збільшити на 1% на кожну одиницю значень загальних характеристик.

Функційний розмір використовується як відносна метрика для порівняння з попередніми проектами, за його допомогою можна обчислити кількість рядків коду, що дозволяє визначити загальну трудомісткість та терміни проекту. Є вільно поширювані інструменти для обчислення функційного розміру [16];

17) *прогнозована оцінка трудовитрат та тривалості проекту - за моделлю Боема* [14, 17] трудовитрати на розробку програмних додатків зростають швидше, ніж розмір додатків. Для представлення даного співвідношення використовується експоненційна функція зі значенням показника, близьким до *1.12*. Тривалість проекту за моделлю Боема зростає експоненційно разом з докладеними до проекту зусиллями. Однак в даному випадку значення експоненти менше 1 і складає близько *0.35*.

Боем оцінив параметри розглядуваних вище співвідношень.

$$\text{Трудовитрати} = a \times \text{KLOC}^b; \quad (14)$$

$$\text{Тривалість} = c \times \text{Трудовитрати}^d = c \times (a \times \text{KLOC}^b)^d = c \times a^d \times \text{KLOC}^{b \times d}, \quad (15)$$

де KLOC – кількість тисяч рядків коду, a, b, c, d – коефіцієнти COCOMO [17].

Для органічного (самостійного) програмного проекту: $a = 2{,}4; b = 1{,}05; c = 2{,}5; d = 0{,}38$.

Для вбудованих програмних проектів (інтеграція апаратного та програмного забезпечення): $a = 3{,}6; b = 1{,}20; c = 2{,}5; d = 0{,}32$. Для проміжних програмних проектів (не органічні, але й не жорстко вбудовані): $a = 3{,}0; b = 1{,}12; c = 2{,}5; d = 0{,}35$.

3. Інтелектуальний метод оцінювання результатів проектування та прогнозування характеристик якості програмного забезпечення (ІМОП)

Суть ІМОП полягає в оцінці результатів проектування та прогнозуванні характеристик якості ПЗ на основі метричного аналізу. ІМОП використовує штучну нейронну мережу (ШНМ), яка здійснює апроксимацію метрик, що характеризують ПЗ на етапі проектування, в результаті чого отримаємо оцінку якості етапу проектування та прогноз характеристик якості розроблюваного ПЗ [2].

На сьогодні, проекти різних типів з різним призначенням та областю застосування вимагають кардинально різних оцінок якості та складності, тобто використання різних типів метрик вже на етапі проектування. Тому для формування вхідних даних ШНМ необхідно провести попереднє опрацювання інформації про тип проекту (множина характеристик проекту $HP = \{hp_k \mid k = 1..n\}$). Цю інформацію в змозі надавати софтверні компанії, зацікавлені в покращенні якості розроблюваного програмного забезпечення та зменшенні його складності. Експертні оцінки впливу типу та області застосування проекту на вибір метрик етапу проектування, надані різними софтверними компаніями, найкраще помістити в централізовану базу знань разом із правилами їх опрацювання для доступу до них інших софтверних компаній.

Також для формування вхідних даних ШНМ знадобляться множина метрик етапу проектування з точними значеннями (множина $TM = \{tm_i \mid i = 1..12\}$) та множина метрик етапу проектування з прогнозованими значеннями (множина $PM = \{pm_j \mid j = 1..17\}$).

Вхідними даними для ШНМ є множина метрик етапу проектування з точним значенням $TMP = \{tmp_a \mid a = 1..r\}$ та множина метрик етапу проектування з прогнозованим значенням $PMP = \{pmp_b \mid b = 1..s\}$, які реагують на проекти певного типу з певними характеристиками, тобто змінюються саме для проекту певного типу і відображають особливості такого проекту.

З одержаних множин TMP, PMP формуються вектори, які подаються на вхід штучної нейронної мережі (ШНМ). Результатом роботи ШНМ є оцінка складності та якості проекту на основі точних метрик етапу проектування та прогноз про складність та якість майбутнього програмного забезпечення на основі прогнозованих метрик етапу проектування. (рис. 1).

Вхідні дані для ШНМ подаються у вигляді множин $TMPV = \{tmpv_i \mid i = 1..12\}$, де $tmpv_i$ - кількісне значення i-ї метрики етапу проектування з точним значенням, якщо ця метрика увійшла до складу множини TMP, інакше нуль (0) та $PMPV = \{pmpv_j \mid j = 1..17\}$, де $pmpv_j$ - кількісне значення j-ї метрики етапу проектування з прогнозованим значенням, якщо ця метрика увійшла до складу множини PMP, інакше нуль (0).

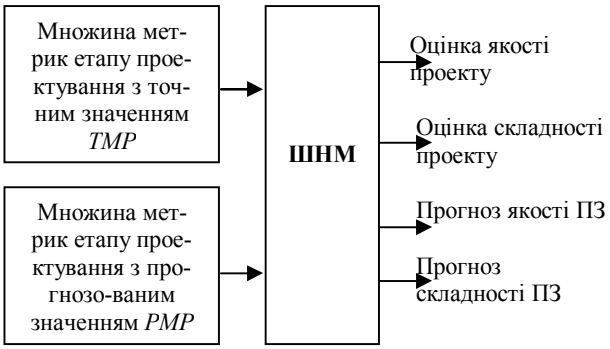

Рис. 1. Нейромережна складова ІМОП

ШНМ для опрацювання метрик етапу проектування ПЗ має 12 входів x' та 17 входів x, оскільки на входи x' подаються кількісні значення метрик етапу проектування з точним значенням, яких в попередньому аналізі виявлено 12, а на входи x подаються кількісні значення метрик етапу проектування з прогнозованим значенням, яких в попередньому аналізі виявлено 17.

На основі кількісних значень елементів множин $TMPV, PMPV$ формуються вхідні вектори для ШНМ.

На вхід x'_i $(i = 1..12)$ подається значення i-го елементу множини $TMPV$, на вхід x_j $(j = 1..17)$ подається j-й елемент множини $PMPV$.

ШНМ опрацьовує набори вхідних векторів та видає вихідні значення:

OQP – оцінка якості проекту в діапазоні [0, 1], де 0 - проект неякісний, а 1 - проект задовольняє вимоги замовника щодо якості;

OSP – оцінка складності проекту в діапазоні [0, 1], де 0 - проект складний для реалізації та передбачає високу вартість реалізації, а 1 - проект простий для реалізації;

PQPZ – прогноз якості ПЗ в діапазоні [0, 1], де 0 – майбутнє ПЗ буде неякісним, а 1 - майбутнє ПЗ очікується високої якості;

PSPZ – прогноз складності ПЗ в діапазоні [0, 1], де 0 – майбутнє ПЗ буде мати суттєву складність, а 1 – майбутнє ПЗ очікується простим, причому дана харктеристика передбачає не тільки простоту чи складність розроблюваного ПЗ з точки зору його обсягу, вартості та часу розробки, але й складність чи простоту супроводу, зручності використання (usability) та ефективність методів, обраних для вирішення задачі.

На основі аналізу 4-х одержаних результатів робиться висновок про якість і складність проекту та очікувану якість і складність розроблюваного за проектом програмного забезпечення.

Висновки

Виходячи з аналізу метрик, значення яких доступні на етапі проектування, можна одержати оцінки, що характеризують етап проектування, який здійснюється конкретною софтверною компанією, і отримати прогнозовані оцінки якості розроблюваного ПЗ за результатами етапу проектування. Одержані оцінки результатів проектування характеризують рівень софтверної компанії та серйозність ставлення компанії до даного замовлення, а також дають дані замовнику для вибору кращої софтверної компанії для розробки необхідного проекту та ПЗ. Прогнозовані оцінки характеристик якості розроблюваного ПЗ дають прогноз щодо якості реалізації конкретної версії проекту та дозволяють порівняти між собою різні версії проекту з такої точки зору.

В ході дослідження автори виявили ряд невирішених задач:

1) типізація проектів та вибір метрик, які реагують на проект певного типу і відображають особливості такого проекту;

2) можлива необхідність розроблення метрик оцінки складності розроблюваного ПЗ з точки зору складність чи простоти його супроводу, зручності використання (usability) та ефективності методів, обраних для вирішення задачі;

3) вибір архітектури штучної нейронної мережі для опрацювання метрик етапу проектування ПЗ.

Література

1. IEEE Standard Glossary of Software Engineering Terminology /IEEE Std 610.12-1990.

2. Поморова О.В. Аналіз методів та засобів оцінки якості програмних систем / О.В. Поморова, Т.О. Говорущенко // Радіоелектронні і комп'ютерні системи. – 2009. – № 6. - С. 148-158.

3. Поморова О.В. Аналіз та опрацювання метрик якості програмного забезпечення на етапі проектування / О.В. Поморова, Т.О. Говорущенко, С.Я. Тарасек // Вісник Хмельницького національного університету. – Хмельницький: ХНУ. - 2010. - № 1. – С. 54-63.

4. Липаев В.В. Выбор и оценивание характеристик качества программных средств: Методы и стандарты / В.В. Липаев - М.: Синтег, 2001. - 224 с.

5. Технологии оценки качества программных продуктов [Электронный ресурс]. – Режим доступа http://www.rol.ru/news/it/press/cwm/25_96/teh.htm.

6. Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем: Учебник для ВУЗов / С.А. Орлов - СПб.: Питер, 2004. - 527 с.

7. Yourdon E. Structured Design: fundamentals of a discipline of computer program and systems design / E. Yourdon, L. Costantine // Englewood Cliffs. - NJ: Prentice-Hall, 1979. - 513 p.

8. Page-Jones M. The Practical Guide to Structured Systems Design / M. Page-Jones // Englewood Cliffs. - NY: Yourdon Pressl, 1988. - 609 p.

9. Модели и метрики оценки качества ПО [Электронный ресурс] – Режим доступа http:// www.met-rix.narod.ru/page2.htm.

10. Новичков А. Метрики кода и их практическая реализация в Subversion и ClearCase. Часть 1 - метрики / А. Новичков, А. Шамрай, А. Черников // http://cmcons.com/articles/CC_CQ/ dev_metrics/mertics_part_1.

11. Петрухин В.А. Методы и средства инженерии программного обеспечения / В.А. Петрухин, Е.М. Лаврищева // [Электронный ресурс] – Режим доступа http://www.intuit.ru/department/se/swebok/ 10/3.html.

12. Construx [Электронный ресурс] – Режим доступа http://www.construx.com.

13. International Function Point User's Group [Электронный ресурс] – Режим доступа http://www. ifpug.org/.

14. Брауде Э. Технология разработки программного обеспечения / Э. Брауде – СПб.: Питер, 2004. - 655 с.

15. Петрухин В.А. Методы и средства инженерии программного обеспечения / В.А. Петрухин, Е.М. Лаврищева // [Электронный ресурс] – Режим доступа http://www.intuit.ru/department/se/swebok/ 10/1.html.

16. International Function Point User's Group reference to function point spread-sheets [Электронный ресурс] – Режим доступа http://ifpug.org/home/docs/ freebies.html.

17. Boehm B. Software Engineering Economics / B. Boehm - NJ: Prentice Hall, 1981. - 392 p.

Надійшла в редакцію 15.01.2010

Рецензент: д-р техн. наук, проф., завідувач кафедри системного програмування В.М. Локазюк, Хмельницький національний університет, Хмельницький.

ИНТЕЛЛЕКТУАЛЬНЫЙ МЕТОД ОЦЕНИВАНИЯ РЕЗУЛЬТАТОВ ПРОЕКТИРОВАНИЯ И ПРОГНОЗИРОВАНИЯ ХАРАКТЕРИСТИК КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

О.В. Поморова, Т.А. Говорущенко

В статье выполнен анализ метрик этапа проектирования с точными и прогнозированными значениями с целью подготовки их к интеллектуальной обработке, на основе которого предложен интеллектуальный метод оценивания результатов проектирования и прогнозирования характеристик качества программного обеспечения (ИМОП).

Ключевые слова: программное обеспечение (ПО), этап проектирования программного обеспечения, метрики качества и сложности программного обеспечения, метрики этапа проектирования ПО, интеллектуальный метод оценивания результатов проектирования и прогнозирования характеристик качества ПО.

INTELLIGENCE METHOD OF SOFTWARE ENGINEERING RESULTS VALUATION AND SOFTWARE QUALITY CHARACTERISTICS PREDICTION

O.V. Pomorova, T.O. Govorushchenko

In article the analysis of design stage metrics with exact and prognosed values for the purpose of theirs preparation for intelligence processing was executed. On basis of this analysis intelligence method of software engineering results valuation and software quality charasteristics prediction (ITVP) was proposed.

Key words: software, software design stage, software quality and complexity metrics, software design stage metrics, intelligence method of software engineering results valuation and software quality charasteristics prediction.

Поморова Оксана Вікторівна – д-р техн. наук, проф., професор кафедри системного програмування, Хмельницький національний університет, м. Хмельницький, Україна, e-mail: o.pomorova@gmail.com.

Говорущенко Тетяна Олександрівна – канд. техн. наук, доцент кафедри системного програмування, Хмельницький національний університет, м. Хмельницький, Україна, e-mail: tat_yana@ukr.net.