УДК 004.891.3: 004.3

А.В. КРАСІЙ
Хмельницький національний університет

НЕЙРОМЕРЕЖНИЙ МЕТОД ПРОГНОЗУВАННЯ ХАРАКТЕРИСТИК ТА ОЦІНЮВАННЯ УСПІШНОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОЕКТУ

Доведено залежність якості та успішності реалізації програмного проекту від специфікації вимог, актуальність та важливість вміння оцінити можливу успішність реалізації програмного проекту на основі специфікації. Вперше запропоновано нейромережний метод прогнозування характеристик та оцінювання успішності реалізації програмного проекту, який полягає у прогнозуванні характеристик програмного проекту на основі аналізу показників специфікації, інтерпретації отриманих відносних значень характеристик програмного проекту за допомогою інтегративного показника проекту, оцінюванні ступеня успішності реалізації програмного проекту на основі інтегративного показника проекту. Запропонований метод відрізняється від відомих тим, що дозволяє прогнозувати успішність реалізації програмних проектів та порівнювати програмні проекти на основі специфікації вимог.

Ключові слова: специфікація вимог до ПЗ, показники специфікації вимог до ПЗ, програмний проект, якість програмного проекту, успішність реалізації програмного проекту, характеристики програмного проекту, вибір програмного проекту, метод прогнозування характеристик та оцінювання успішності реалізації програмного проекту (МПХОУР), інтегративний показник програмного проекту, ступінь успішності реалізації програмного проекту.

A.V. KRASIY
Khmelnitskiy National University

NEURONET METHOD OF CHARACTERISTICS PREDICTION AND SOFTWARE PROJECT IMPLEMENTATION SUCCESS EVALUATION

Necessity to deepen the analysis of software requirements specification (SRS), dependence of quality and success of software project implementation from SRS, actuality and importance of skill to evaluate the possible success of software project based on SRS and the need to support the developer and customer in choosing the software project were proved. The neuronet method of characteristics prediction and software project implementation success evaluation was first proposed. It consists of next stages: neuronet prediction of characteristics of software project based on the analysis of specification, interpretation of the received relative values of the software project characteristics on the basis of integrative indicator of project, evaluation of the degree of success of the software project implementation based on the integrative indicator of project, testing of the stability and acceptability of compensations of characteristics of software project This method differs from the known methods that provides the prediction of success of software projects implementation and comparison of software projects on the basis of SRS.

Keywords: software requirements specification (SRS), SRS indicators, software project, software project quality, success of software project implementation, software project characteristics, the choice of software project, method of characteristics prediction and software project implementation success evaluation, integrative indicator of software project, the degree of success of the software project implementation.

Вступ

Ключовим фактором забезпечення ефективного застосування програмного забезпечення (ПЗ) є ретельне оцінювання та досягнення високих значень показників якості.

Аналітичні дослідження та огляди, які виконували протягом кількох останніх років провідні зарубіжні аналітики, давали не дуже обнадійливі результати в галузі забезпечення якості програмних продуктів. За статистичними оцінками, наведеними у [1], більше 50 % від загальної кількості корпоративних програмних проектів не виправдовують очікувань, причому роботи над 42 % проектів припиняються задовго до їх логічного завершення. Дослідження Standish Group, в ході яких було проаналізовано роботу 364 американських софтверних корпорацій і підсумки виконання понад 23 тисяч проектів по розробленню ПЗ, показало, що проекти вартістю менше 750 тисяч доларів виявляються успішними у 55% випадків, вартістю - від 1 до 2 млн - у 18 %, вартістю від 5 до 10 мільйонів - всього у 7 % випадків [1]. Статистика успішності реалізації програмних проектів за 1994-2012 роки, за даними The Standish Group International (Chaos reports) [2], представлена на рисунку 1. Успішними вважаються проекти, які були виконані вчасно, в рамках бюджету, з необхідними можливостями та функціями; проблемними вважаються проекти, які мали перевищення термінів, перевитрати або не мали необхідних можливостей та функцій; провальними (невдалими) вважаються проекти, які були скасовані до завершення, або були виконані, але ніколи не використовувались.

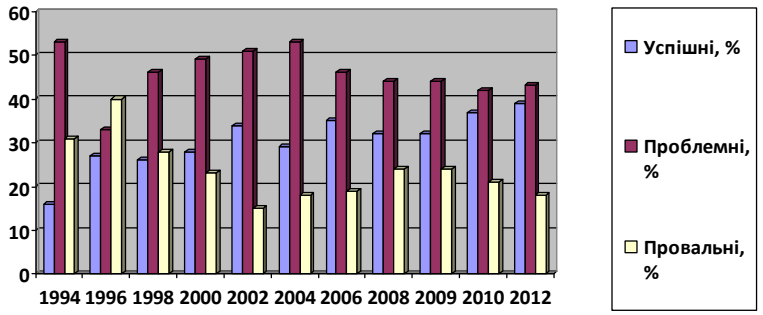

Рис. 1. Статистика успішності реалізації програмних проектів у 1994-2012 рр

Аналіз даних рис.1 дав можливість побачити приріст кількості успішних проектів та спад кількості провальних проектів у 2010-2012 роках, але в той же час частка проблемних проектів є досить сталою величиною у 2006-2012 роках і складає 42-46% проектів. Крім цього, на рис.1 можна побачити своєрідну періодичність (синусоїдальність) приростів та спадів кількостей для всіх груп проектів, тому не виключено, що в наступні роки може відбутись спад кількості успішних проектів і приріст кількості провальних проектів.

Статистика по перевитратах, перевищенню термінів та відсутності необхідної функціональності для проблемних програмних проектів у 2004-2012 роках, за даними The Standish Group International (CHAOS reports) 2013 року [2], представлена на рисунку 2.

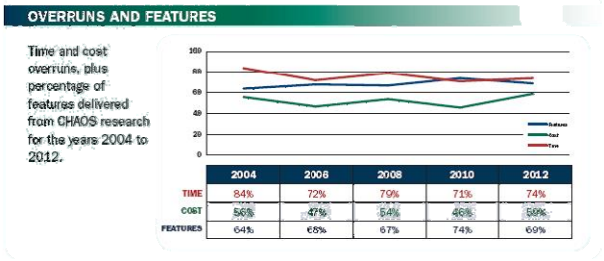

Рис. 2. Статистика по перевитратах, перевищенню термінів та відсутності необхідної функціональності для проблемних програмних проектів у 2004-2012 роках

Наведені статистичні дані фактично відображають досить високу частку неякісних програмних проектів – від 84% у 1994 році до 61% у 2012 році, оскільки не тільки провальні, але й проблемні програмні проекти не можуть вважатись якісними з точки зору стандартного трактування поняття якості ПЗ (як ступеня відповідності характеристик ПЗ вимогам [3-5]).

Велика кількість помилок ПЗ виникає на етапі формулювання вимог. Як показано у [1], помилки формулювання вимог складають 10-23% всіх помилок, причому чим більший обсяг ПЗ, тим більше помилок вноситься саме при формулюванні вимог. Аналіз помилок вбудованого та прикладного ПЗ, що призвели до відомих катастроф та інцидентів та були внесені на етапі формулювання вимог наведено у таблиці 1.

Таблиця 1

Аналіз помилок етапу формулювання вимог вбудованого і прикладного ПЗ та їхніх наслідків

Подія	Причина	Наслідки
1	2	3
Вбудоване ПЗ		
У 1971 р. "Космос-419" не стартував до Марсу [6]	Невірність специфікації	Втрата апарату
У 1971 р. станція "Марс 2" не змогла відстикуватись від корабля [6]	Невірність специфікації	Невиконання завдання
1	2	3
Аварія гелікоптера Chinook у 1994 році [6]	Невірність специфікації	Загинуло 29 чоловік
"Смертельні" сеанси радіаційної терапії із застосуванням Therac-25 [7, 8]	Неповнота специфікації	6 пацієнтів одержали смертельну дозу опромінення
Вибух ракети-носія Ariane 5 у1996 році [7, 8]	Протиріччя вимог	Вартість обладнання - 7,5 млрд. доларів, "упущена вигода" – 2 млрд. доларів
Прикладне ПЗ		
Падіння рейтингу на біржових торгах компанії Dow Jones Industrial Average (1987 р.) [7, 8]	Невірний розрахунок навантаження на ПЗ	Втрата 500 мільярдів доларів
Збій у системі Нью-Йоркського банку [9]	Недостатність пам'яті через невірні вимоги	32 млрд. доларів
Помилка у грі «Король Лев» компанії «Дісней» (1994 р.) [10]	Неповнота специфікації	Втрата репутації, значні фінансові втрати компанії
Помилка Y2K - помилка двоцифрового збереження року у даті (1999 р.) [7, 8]	Невірність або неповнота специфікації	Втрати - 500 млрд. доларів
У 1990 р. на AT&T відбулась 9-годинна аварія [8]	Проблеми з граничними умовами у специфікації	75 млн. нездійснених дзвінків, втрата 60 млн. доларів
У 1998 р. на AT&T відбулась 26-годинна аварія [7]	Проблеми з прихованими граничними умовами у специфікації	Непрацездатність служб, пов’язаних з передачею даних

В роботах [11-13] підтверджується факт, що причини майже всіх інцидентів та катастроф, пов’язаних з ПЗ, криються у специфікації вимог до програмного забезпечення, тобто переважна більшість

аварій, пов'язаних із ПЗ, виникли через помилкові вимоги, а не через помилки кодування. У [12] описано результати експерименту, проведеного для підтвердження або відкидання гіпотези про те, що збої та помилки ПЗ, написаного різними розробниками за однією специфікацією, статистично незалежні. Під час експерименту декілька незалежних груп розробників писали свою версію ПЗ за однією специфікацією. В результаті такого експерименту було встановлено, що версії ПЗ, написані різними розробниками за однаковими вимогами, містили ряд спільних помилок, пов'язаних із помилками або неточностями вимог (специфікації). Аналіз великої кількості програмних проектів, проведений у [14], також виявив, що головне місце виникнення помилок у ПЗ – це етап формулювання вимог (специфікація). Робота [15] основними причинами провалу програмних проектів називає невірні уявлення керівництва та мене-джерів проекту щодо реального часу та коштів, які необхідні для забезпечення функційних вимог користувача, тобто знов-таки підтверджує, що більшість проблем ПЗ пов'язана із проблемами специфікації.

Отже, при сучасному визначенні якості ПЗ [3-5] – якщо встановлені цілі проекту не відповідають потребам користувачів, то програмний продукт неможливо буде визнати якісним, навіть якщо при його розробленні були використані сучасні технології та були задіяні найкваліфікованіші розробники. Тоді якість та успішність реалізації програмного проекту залежать від специфікації вимог. Таке експериментальне свідчення безпосередньо призводить до необхідності поглиблення аналізу специфікації. Крім цього, проілюстроване на рисунку 2 зниження кількості проектів із відсутньою необхідною функціональністю при зростанні кількості проектів із перевищенням термінів та перевитратами також може відображати тенденцію, що софтверні організації поглиблюють аналіз та опрацювання вимог користувача за рахунок збільшення часу та коштів.

У презентації [16] є слайд, який показує, що частка вимог-фактів є суттєво меншою, ніж вимог – необґрунтованих припущень.

Тоді наразі *актуальним* і дуже важливим є аналіз специфікації вимог до ПЗ, вміння оцінити можливу успішність реалізації програмного проекту на основі специфікації, а також підтримка розробника та замовника при виборі програмного проекту з множини альтернативних програмних проектів (наразі розробник та замовник керуються при здійсненні такого вибору лише вартістю та тривалістю проекту, а також власною інтуїцією). Враховуючи вищевикладене, *метою даного дослідження* є прогнозування характеристик та оцінювання успішності реалізації програмного проекту на основі аналізу специфікації вимог до ПЗ.

Визначення 1. Під *успішністю реалізації програмного проекту* надалі будемо розуміти вчасне виконання програмного проекту в рамках виділеного бюджету та з реалізацією всіх необхідних можливостей та функцій.

Успішність реалізації програмного проекту на етапі проектування можна ймовірнісно оцінити на основі прогнозованих значень основних характеристик програмного проекту [17] – тривалості проекту, вартості, складності, кросплатформності, зручності використання, якості. Аналіз зазначених характеристик [18, 19] зробив очевидним факт, що існуючі математичні інструменти та методи, а також автоматизовані засоби їх визначення не придатні для оцінки їх значень на етапі формулювання вимог, оскільки орієнтовані на готовий програмний код, а не на наявну специфікацію вимог до ПЗ.

Дослідження відомих методів (Using natural language processing technique, Using CASE analysis method, QAW-method, Using global analysis method, O'Brien's approach, Method to discov-er missing requirement elicitation, Selection of elicitation technique, Comparison and categorization of requirements elicitation techniques, Techniques for ranking and prioritization of software requirements) та автоматизованих засобів (OSRMT, Tools by LDRA, Sigma Software, DEVPROM, CASE.Analytics) аналізу специфікацій [18-22], призначених для роботи з вимогами та специфікаціями, показало, що всі вони спрямовані на контроль за реалізацією вимог, але жоден з них не визначає прогнозованих значень характеристик за специфікацією. Отже, існуючі методи і засоби аналізу специфікацій не прийнятні для кількісного оцінювання характеристик програмного проекту на основі аналізу специфікацій. Тоді для прогнозування успішності реалізації програмного проекту на основі аналізу специфікації слід вирішити *задачу* розроблення методу прогнозування характеристик та оцінювання успішності реалізації програмного проекту на основі аналізу специфікації вимог.

Нейромережний метод прогнозування характеристик та оцінювання успішності реалізації програмного проекту (МПХОУР)

Аналіз специфікації вимог до ПЗ [23, 24] показав, що вимоги специфікації дозволяють сформувати множини показників, на основі яких замовник і розробник можуть отримати прогнозовані кількісні значення характеристик програмного проекту, що дозволяють отримати прогноз успішності реалізації даного програмного проекту: $R1$ - множина показників розділу 1 специфікації вимог до ПЗ, $R2$ - множина показників розділу 2, $R3$ - множина показників розділу 3, $R4$ - множина показників розділу 4 специфікації вимог до ПЗ. Показники специфікації, які входять до множин $R1 - R4$, детально розглядаються у [25].

Тоді метод прогнозування характеристик та оцінювання успішності реалізації програмного проекту на основі аналізу специфікації вимог складається з наступних етапів:

1) нейромережне прогнозування характеристик програмного проекту на основі аналізу специфікації;

2) інтерпретація отриманих відносних значень характеристик програмного проекту – критерієм для

такої інтерпретації є інтегративний показник проекту;

3) оцінювання ступеня успішності реалізації програмного проекту на основі інтегративного показника проекту;

4) перевірка стабільності та припустимості компенсаційних впливів характеристик програмного проекту.

Першим етапом МПХОУР є прогнозування характеристик програмного проекту на основі аналізу специфікації, що полягає в опрацюванні зазначених множин показників $R1-R4$ та у визначенні відносних значень основних характеристик програмного проекту: *Cs* – вартість програмного проекту, *Dsp* – тривалість, *Cp* – кросплатформність, *Cx* – складність, *Ub* – зручність використання, *Qs* – якість. Наразі відсутні функції (формули, залежності), за якими можна обчислити значення тієї чи іншої характеристики ПЗ на основі множини впливових показників специфікації - всі наявні формули та методики оцінювання характеристик ПЗ орієнтовані на готовий програмний код, а не на специфікацію вимог. Теорема Хехт-Нільсена доводить можливість розв'язку задачі представлення багатовимірної функції довільного вигляду на штучній нейронній мережі (ШНМ), тому для реалізації невідомих функцій залежності характеристик ПЗ від показників специфікації використовуватимемо саме ШНМ. У [26] доведено необхідність, математично описано та розроблено штучну нейронну мережу (ШНМ), яка опрацьовує множину показників специфікації, здійснює апроксимацію показників та надає прогнозовані кількісні значення характеристик ПЗ.

ШНМ прогнозування характеристик програмного проекту на основі аналізу специфікації навчена так, що всі значення характеристик належать інтервалу (0;1], причому значення кожної характеристики, близьке до 0, негативно впливає на успішність реалізації програмного проекту (високі вартість, тривалість, складність; низькі якість, кросплатформність та зручність використання), а значення кожної характеристики, близьке до 1, позитивно впливає на успішність реалізації проекту (низькі вартість, тривалість, складність; високі якість, кросп-латформність та зручність використання).

На основі отриманих з ШНМ відносних значень основних характеристик проекту як розробнику, так і замовнику складно комплексно оцінити успішність реалізації програмного проекту, оскільки складно вірно інтерпретувати одержані відносні значення характеристик, тому *другим етапом МПХОУР* є інтерпретація отриманих відносних значень характеристик програмного проекту. Для інтерпретації відносних значень характеристик введемо поняття інтегративного показника проекту.

Визначення 2. *Інтегративний показник* Iip_{Sp} програмного проекту *Sp* – це кількісний показник успішності реалізації програмного проекту на основі множини відносних значень прогнозованих характеристик проекту.

Для визначення впливу характеристик проекту на інтегративний показник проекту та взаємозв'язків між характеристиками відсутні формули та залежності. Тому припустимо, що всі шість зазначених характеристик мають однаковий вплив на успішність реалізації програмного проекту, а відтак і на ітегративний показник проекту. За відсутності формул та залежностей найбільш простим та очевидним способом визначення інтегративного показника проекту є використання його графічного представлення. Для отримання графічного представлення Iip_{Sp} створимо систему координат, яка має шість основних осей (для шести характеристик проекту) – рисунок 3.

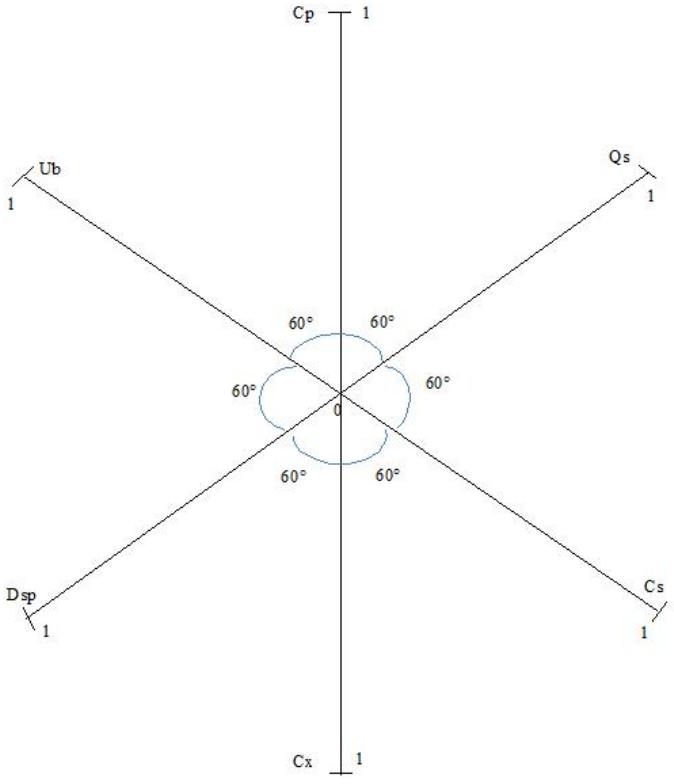

Рис. 3. Система координат для інтегративного показника Iip_{Sp} проекту

Тоді інтегративний показник проекту – це площа фігури, сформованої прогнозованими відносними значеннями характеристик проекту. Оскільки ШНМ надає значення шести характеристик (Cs_{ANN}, Dsp_{ANN}, Cp_{ANN}, Cx_{ANN}, Ub_{ANN}, Qs_{ANN}), то інтегративний показник проекту – це площа шестикутника $Cs_{ANN}Cx_{ANN}Dsp_{ANN}Ub_{ANN}Cp_{ANN}Qs_{ANN}$, окресленого жирною лінією на рисунку 4.

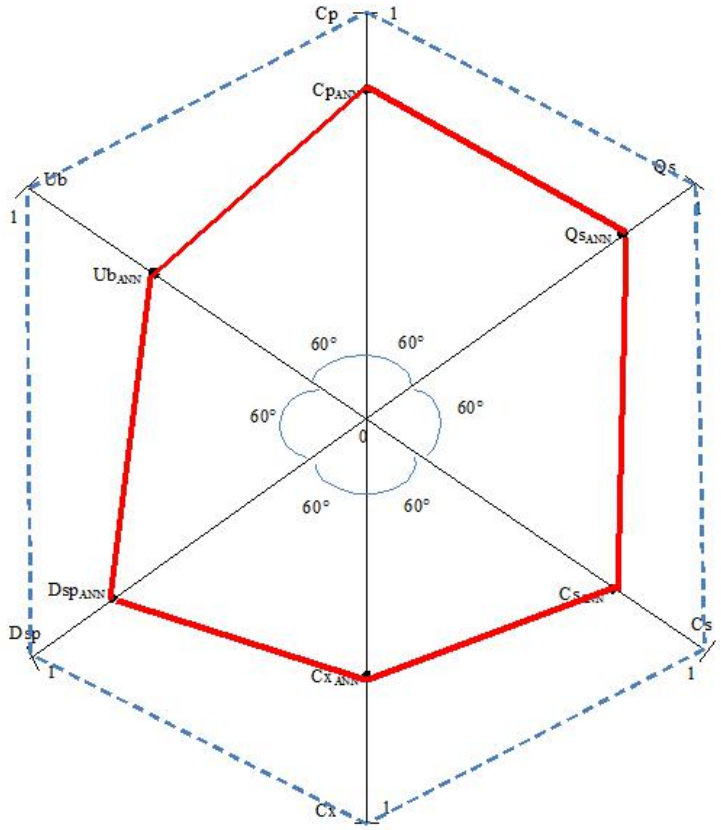

Рис.4 – Графічне представлення інтегративного показника Iip_{Sp} проекту та його максимального значення $Iip_{\max}$

Для знаходження інтегративного показника Iip_{Sp} проекту розіб'ємо шестикутник на шість трикутників - $Cs_{ANN}OCx_{ANN}$, $Cx_{ANN}ODsp_{ANN}$, $Dsp_{ANN}OUb_{ANN}$, $Ub_{ANN}OCp_{ANN}$, $Cp_{ANN}OQs_{ANN}$, $Qs_{ANN}OCs_{ANN}$ і знайдемо площу для кожного з трикутників за двома відомими сторонами (значення характеристик) та кутом між ними (кут дорівнює 60° згідно побудованої системи координат), після чого додамо отримані площі трикутників:

$$S_{Cs_{ANN}OCx_{ANN}} = \frac{1}{2}\cdot Cs_{ANN}\cdot Cx_{ANN}\cdot \sin 60^\circ = 0.5\cdot Cs_{ANN}\cdot Cx_{ANN}\cdot 0.866 = 0.433\cdot Cs_{ANN}\cdot Cx_{ANN} \quad (1)$$

$$S_{Cx_{ANN}ODsp_{ANN}} = \frac{1}{2}\cdot Cx_{ANN}\cdot Dsp_{ANN}\cdot \sin 60^\circ = 0.433\cdot Cx_{ANN}\cdot Dsp_{ANN} \quad (2)$$

$$S_{Dsp_{ANN}OUb_{ANN}} = \frac{1}{2}\cdot Dsp_{ANN}\cdot Ub_{ANN}\cdot \sin 60^\circ = 0.433\cdot Dsp_{ANN}\cdot Ub_{ANN} \quad (3)$$

$$S_{Ub_{ANN}OCp_{ANN}} = \frac{1}{2}\cdot Ub_{ANN}\cdot Cp_{ANN}\cdot \sin 60^\circ = 0.433\cdot Ub_{ANN}\cdot Cp_{ANN} \quad (4)$$

$$S_{Cp_{ANN}OQs_{ANN}} = \frac{1}{2}\cdot Cp_{ANN}\cdot Qs_{ANN}\cdot \sin 60^\circ = 0.433\cdot Cp_{ANN}\cdot Qs_{ANN} \quad (5)$$

$$S_{Qs_{ANN}OCs_{ANN}} = \frac{1}{2}\cdot Qs_{ANN}\cdot Cs_{ANN}\cdot \sin 60^\circ = 0.433\cdot Qs_{ANN}\cdot Cs_{ANN} \quad (6)$$

$$Iip_{Sp} = 0.433\cdot(Cs_{ANN}\cdot Cx_{ANN} + Cx_{ANN}\cdot Dsp_{ANN} + Dsp_{ANN}\cdot Ub_{ANN} + Ub_{ANN}\cdot Cp_{ANN} + Cp_{ANN}\cdot Qs_{ANN} + Qs_{ANN}\cdot Cs_{ANN}) \quad (7)$$

Порядок осей шестикутника було обрано з врахуванням особливостей навчання ШНМ та з міркувань неможливості компенсації одних характеристик іншими (оскільки всі вищезазначені шість характеристик є рівно важливими для програмного проекту). З формули (7) видно, що при множенні значення однієї характеристики на значення іншої може відбуватись компенсація низького значення однієї характеристики високим значенням іншої характеристики. Тому верхня частина системи координат складається з трьох осей для характеристик Ub, Cp, Qs, а нижня частина складається з трьох осей для характеристик Cs, Cx, Dsp, для яких діє правило навчання ШНМ: значення характеристики, близьке до 0, означає високі вартість, тривалість, складність, низькі якість, кросплатформність та зручність використання. Стикування осей підібрано попарно саме так, тому що не повинна низька ціна програмного проекту ($Cs \to 1$) компенсувати його низьку якість ($Qs \to 0$), низька тривалість програмного проекту ($Dsp \to 1$) не може компенсувати його низьку зручність використання ($Ub \to 0$). Перевірка припустимості компенсацій характеристик здійснюватиметься на четвертому етапі методу, і саме такий порядок осей дозволить виявити

неприпустимі компенсації значень характеристик.

Для подальшої роботи знадобиться також максимальне значення інтегративного показника проекту: $Iip_{\max}$ - площа шестикутника $CsCxDspUbCpQs$, окресленого пунктирною лінією на рисунку 4. ШНМ навчена так, що максимально можливе значення кожної характеристики – це 1. Тоді:

$$IIp_{\max} = 0.433 \cdot (Cs \cdot Cx + Cx \cdot Dsp + Dsp \cdot Ub + Ub \cdot Cp + Cp \cdot Qs + Qs \cdot Cs) = \\ = 0.433 \cdot (1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1) = 2.598 \quad (8)$$

Сам по собі інтегративний показник проекту є неінформативним для розробника та замовника через складність інтерпретації його значення, тому *третім етапом МПХОУР* є оцінювання ступеня успішності реалізації програмного проекту. Інтегративний показник програмного проекту дає змогу визначити ступінь успішності реалізації програмного проекту. Значення $Iip_{\max} = 2.598$ – це найкраще значення інтегративного показника проекту, тоді ступінь P_{Iip} успішності реалізації програмного проекту складає:

$$P_{Iip} = \frac{Iip_{Sp}}{Iip_{\max}} = \frac{Iip_{Sp}}{2.598} = 0.385 \cdot Iip_{Sp} \quad (9)$$

Значення ступеня успішності реалізації програмного проекту, близьке до 0, показує низьку успішність реалізації програмного проекту (такий проект найімовірніше буде провальним), а значення ступеня успішності реалізації програмного проекту, близьке до 1, показує високу успішність реалізації програмного проекту, тобто проект з таким ступенем успішності буде успішним (оптимізаційна задача). Формалізація першого – третього етапів МПХОУР матиме наступний вигляд:

$$< R1, R2, R3, R4 > \Rightarrow < Cs_{ANN}, Cx_{ANN}, Dsp_{ANN}, Ub_{ANN}, Cp_{ANN}, Qs_{ANN} >= f_1(< R1, R2, R3, R4 >) \Rightarrow \\ \Rightarrow Iip_{Sp} = f_2(< Cs_{ANN}, Cx_{ANN}, Dsp_{ANN}, Ub_{ANN}, Cp_{ANN}, Qs_{ANN} >) \Rightarrow P_{Iip} = f_3(Iip_{Sp}, Iip_{\max}) \quad (10)$$

,

де функція f_1 реалізується навченою штучною нейронною мережею, функція f_2 обчислюється за формулою (7), функція f_3 – за формулою (9); показник $Iip_{\max}$ обчислюється за формулою (8).

МПХОУР відрізняється від відомих тим, що дозволяє прогнозувати успішність реалізації програмних проектів, порівнювати програмні проекти комплексно за основними характеристиками та прогнозованим значенням ступеня успішності реалізації програмних проектів (а не тільки за вартістю та тривалістю, як відбувається наразі) та виконувати обґрунтований вибір програмного проекту розробником та замовником для подальшої реалізації. Звісно, помилки у ПЗ можуть бути внесені і на наступних етапах – під час проектування та програмування, але даний метод допомагає «відсікти» програмні проекти з невдалою специфікацією, оскільки, як доведено вище, програмні проекти з невдалими вимогами та специфікаціями не можуть мати успішної реалізації.

Як відомо, значення основних характеристик програмного проекту можуть коливатись в діапазоні (0;1]. Зрозуміло, що не завжди характеристики матимуть приблизно однакові значення. Тоді очевидно, що низькі значення одних характеристик можуть бути скомпенсовані високими значеннями інших характеристик при обчисленні інтегративного показника програмного проекту. Така компенсація значень характеристик при отриманні однакових значень інтегративного показника не завжди є коректною, оскільки всі вищезазначені шість характеристик є рівно важливими для програмного проекту, тому не повинні компенсуватись одна одною.

Тоді *четвертим етапом МПХОУР* є перевірка стабільності та припустимості компенсаційних впливів характеристик програмного проекту. Як було зазначено вище, інтегративний показник проекту – це площа виділеного суцільною жирною лінією шестикутника $Cs_{ANN}Cx_{ANN}Dsp_{ANN}Ub_{ANN}Cp_{ANN}Qs_{ANN}$. Якщо зазначений шестикутник буде випуклим, то характеристики програмного проекту вважатимуться стабільними, а їхні компенсаційні впливи – припустимими.

Введемо *показник* Ace_{Sp} *стабільності та припустимості компенсаційних впливів характеристик*, який прийматиме значення *True*, якщо шестикутник випуклий, тобто характеристики стабільні, їхні компенсаційні впливи припустимі, і значення *False*, якщо шестикутник не є випуклим (характеристики не стабільні).

Критерієм випуклості шестикутника, як відомо, є одночасне виконання двох умов: 1) однаковий знак синусів усіх кутів шестикутника; 2) сума всіх кутів випуклого шестикутника складає 720° (за теоремою про суму кутів випуклого багатокутника) – для перевірки, що шестикутник – не зірка (не пентаграма). Наведемо формули для визначення кутів шестикутника (за рис.4):

1) за теоремою косинусів обчислимо невідому третю сторону для кожного вищезазначеного трикутника:

$$CpQs^2 = Cp^2 + Qs^2 - 2 \cdot Cp \cdot Qs \cdot \cos 60^\circ \quad (11)$$

$$QsCs^2 = Qs^2 + Cs^2 - 2 \cdot Qs \cdot Cs \cdot \cos 60^\circ \quad (12)$$

$$CsCx^2 = Cs^2 + Cx^2 - 2 \cdot Cs \cdot Cx \cdot \cos 60^\circ \quad (13)$$

$$CpDsp^2 = Cx^2 + Dsp^2 - 2 \cdot Cx \cdot Dsp \cdot \cos 60^\circ \quad (14)$$

$$DspUb^2 = Dsp^2 + Ub^2 - 2 \cdot Dsp \cdot Ub \cdot \cos 60^\circ \quad (15)$$

$$UbCp^2 = Ub^2 + Cp^2 - 2 \cdot Ub \cdot Cp \cdot \cos 60^\circ \quad (16)$$

2) за теоремою косинусів знайдемо один з двох невідомих кутів у кожному трикутнику:

$$\angle(OQsCp) = \arccos\left(\frac{[Qs^2 + CpQs^2 - Cp^2]}{2 \cdot Qs \cdot CpQs}\right) \quad (17)$$

$$\angle(OCsQs) = \arccos\left(\frac{[Cs^2 + QsCs^2 - Qs^2]}{2 \cdot Cs \cdot QsCs}\right) \quad (18)$$

$$\angle(OCxCs) = \arccos\left(\frac{[Cx^2 + CsCx^2 - Cs^2]}{2 \cdot Cx \cdot CsCx}\right) \quad (19)$$

$$\angle(ODspCx) = \arccos\left(\frac{[Dsp^2 + CxDsp^2 - Cx^2]}{2 \cdot Dsp \cdot CxDsp}\right) \quad (20)$$

$$\angle(OUbDsp) = \arccos\left(\frac{[Ub^2 + DspUb^2 - Dsp^2]}{2 \cdot Ub \cdot DspUb}\right) \quad (21)$$

$$\angle(OCpUb) = \arccos\left(\frac{[Cp^2 + UbCp^2 - Ub^2]}{2 \cdot Cp \cdot UbCp}\right) \quad (22)$$

3) за теоремою про суму кутів трикутника знайдемо другий невідомий кут у кожному трикутнику:

$$\angle(OCpQs) = 180^\circ - 60^\circ - \angle(OQsCp) \quad (23)$$

$$\angle(OQsCs) = 180^\circ - 60^\circ - \angle(OCsQs) \quad (24)$$

$$\angle(OCsCx) = 180^\circ - 60^\circ - \angle(OCxCs) \quad (25)$$

$$\angle(OCxDsp) = 180^\circ - 60^\circ - \angle(ODspCx) \quad (26)$$

$$\angle(ODspUb) = 180^\circ - 60^\circ - \angle(OUbDsp) \quad (27)$$

$$\angle(OUbCp) = 180^\circ - 60^\circ - \angle(OCpUb) \quad (28)$$

4) знайдемо тепер кути шестикутника:

$$\angle(Qs) = \angle(OQsCp) + \angle(OQsCs) \quad (29)$$

$$\angle(Cs) = \angle(OCsQs) + \angle(OCsCx) \quad (30)$$

$$\angle(Cx) = \angle(OCxCs) + \angle(OCxDsp) \quad (31)$$

$$\angle(Dsp) = \angle(ODspCx) + \angle(ODspUb) \quad (32)$$

$$\angle(Ub) = \angle(OUbDsp) + \angle(OUbCp) \quad (33)$$

$$\angle(Cp) = \angle(OCpUb) + \angle(OCpQs) \quad (34)$$

Після знаходження кутів шестикутника слід знайти синуси кутів, отриманих за формулами (29)-(34), і порівняти їх знаки, а також знайти суму кутів, отриманих за формулами (29)-(34), і перевірити, чи дорівнює ця сума 720°. Якщо сума кутів дорівнює 720°, а синуси кутів мають однакові знаки, то шестикутник є випуклим, відповідно показник стабільності та припустимості компенсаційних впливів характеристик $Ace_{Sp} = True$ (характеристики програмного проекту є стабільними, а їх компенсаційні впливи – припустимими).

Висновки

У статті доводиться необхідність поглиблення аналізу специфікації, залежність якості та успішності реалізації програмного проекту від специфікації вимог, актуальність та важливість вміння оцінити можливу успішність реалізації програмного проекту на основі специфікації, а також необхідність підтримки розробника та замовника при виборі програмного проекту з множини альтернативних програмних проектів (наразі розробник та замовник керуються при здійсненні такого вибору лише вартістю та тривалістю проекту, а також власною інтуїцією).

Автором вперше запропоновано нейромережний метод прогнозування характеристик та оцінювання успішності реалізації програмного проекту. МПХОУР полягає у:

1) нейромережному прогнозуванні характеристик програмного проекту на основі аналізу показників специфікації;

2) інтерпретації отриманих відносних значень характеристик програмного проекту за допомогою інтегративного показника проекту;

3) оцінюванні ступеня успішності реалізації програмного проекту на основі інтегративного

показника проекту;

4) перевірці стабільності та припустимості компенсаційних впливів характеристик програмного проекту.

Запропонований метод відрізняється від відомих тим, що дозволяє прогнозувати успішність реалізації програмних проектів на основі лише специфікації вимог та порівнювати програмні проекти комплексно – за прогнозованими значеннями основних характеристик та значеннями ступеня успішності реалізації програмного проекту (а не тільки за вартістю та тривалістю, як відбувається наразі). Практичне значення розробленого методу полягає у його використанні для виконання обґрунтованого вибору програмного проекту замовником і розробником для подальшої реалізації.

Перспективою для подальших досліджень авторів є:

1) підвищення достовірності роботи даного методу за рахунок підвищення достовірності роботи ШНМ прогнозування основних характеристик програмного проекту на основі аналізу специфікацій, для чого необхідним є збір великої кількості показників зі специфікацій вже розроблених проектів та значень характеристик розробленого ПЗ для побудови навчальної вибірки ШНМ;

2) підбір варіативного (диверсного) компоненту, який здійснюватиме прогнозування основних характеристик ПЗ на основі аналізу специфікацій;

3) розроблення програмних засобів (інформаційної технології) прогнозування характеристик та оцінювання успішності реалізації програмного проекту на основі аналізу специфікацій, яка підтримуватиме: збір показників специфікації, опрацювання показників специфікації за допомогою ШНМ, збір прогнозованих відносних значень характеристик програмного проекту, розрахунок інтегративного показника проекту та ступеня успішності реалізації програмного проекту, перевірку стабільності та припустимості компенсаційних впливів характеристик програмного проекту.

Література

1. S.McConnell. Code complete. – Microsoft Press, 2013. – 896 p.

2. CHAOS Manifesto: Think big, act small – The Standish Group International: CHAOS Knowledge Center, 2013 – 52 p.

3. ISO 9000:2005 Quality management systems – Fundamentals and vocabulary

4. ISO 9001:2008 Quality management systems – Requirements

5. P.Bourque. Guide to the software engineering body of knowledge (SWEBOK). Version 3.0 / Bourque P., R.E.Fairley. – A project of the IEEE Computer Society, 2014. – 335 p.

6. А.В.Турчин. Структура глобальной катастрофы: Риски вымирания человечества в XXI веке / А.В. Турчин. – М.: Изд-во ЛКИ, 2011. – 432 с.

7. Software goes wrong, we all know that, but just how wrong can it go? // [Electronic resource] – Access mode: http://www.datareservoir.co.uk/bugs /

8. 20 famous software disasters // [Electronic resource] – Access mode: http://sandipsandilya.wordpress.com/2011/01/17/20-famous-software-disasters/

9. Explaining settlement fails // [Electronic resource] – Access mode: http://www.ny.frb.org/research/current_issues/ci11-9/ci11-9.html

10. R.Patton. Software testing: 2nd edition / Patton R. — Indianapolis: Sams, 2005. — 408p.

11. N.G.Levenson. Systemic factors in software-related spacecraft accidents / Levenson N.G.// AIAA Space 2001 Conference and Exposition, pp.1-11

12. N.G.Leveson Software challenges in achieving space safety / Levenson N.G.// Journal of the British Interplanetary Society - Vol. 62, July/August 2009, pp. 265-272

13. T.Ishimatsu. Hazard analysis of complex spacecraft using systems-theoretic process analysis / Ishimatsu T., Leveson N.G., Thomas J.P., Fleming C.H., Katahira M., Miyamoto Yu., Ujiie R., Nakao H., Hoshino N. // Journal of Spacecraft and Rockets - Vol. 51, No. 2 (2014), pp. 509-522

14. C.Jones. The economics of software quality / Jones C., Bonsignour O. – Boston: Pearson Education, 2012. – 588 p.

15. E.Yourdon. Death March: The complete software developer's guide to surviving "Mission impossible" projects (2nd edition) / Yourdon E.– Prentice Hall, 200

16. N.Johnson. Software testing: Innovate or die // [Electronic resource] – Access mode: http://www.slideshare.net/johnsonnigel/testing-innovate-or-die-42187102

17. A.Maedche. Software for people: fundamentals, trends and best practices (Management for professionals) / Maedche A., Botzenhardt A., Neer L. – Berlin: Springer-Verlag Berlin Heidelberg, 2012. – 293 p.

18. Красій А.В.. Моделювання процесу прогнозування характеристик програмного забезпечення на основі аналізу специфікацій / А.В. Красій // Науковий журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництво". – Луцьк: Луцький національний технічний університет, 2014. – С.66-76

19. N.Fenton. Software metrics: A rigorous approach (3rd edition) – CRC Press, 2014 – 597 p.

20. A.Chen. Visual models for software requirements / Chen A., Beatty J. – W.: MS Press, 2012. – 444 p.

21. A.Fatwanto. Software requirements specification analysis using natural language processing technique / Fatwanto A. // International Conference on Quality in Research – 2013, pp.105-110

22. T.Rehman. Analysis of requirement engineering processes, tools/techniques and methodologies / Rehman T., Khan M.N.A., Riaz N. // I.J. Information Technology and Computer Science. – 2013. – pp.40-48

23. IEEE 830-1998. Recommended practice for software requirements specifications

24. K.Wiegers. Software requirements: 3-rd edition / Wiegers K., Beatty J. – W.: MS Press, 2013. – 640 p.

25. Говорущенко Т.О.. Математичне моделювання специфікації вимог та характеристик програмного забезпечення / Т.О. Говорущенко А.В. Красій // Радіоелектронні і комп'ютерні системи. – Харків: НАУ "ХАІ", 2014. – № 5, – С.34-39

26. Красій А.В.. Нейромережна модель прогнозування характеристик програмного забезпечення на основі аналізу специфікацій / А.В. Красій // Інтелектуальні технології в системному програмуванні. III Всеукраїнська науково-практична конференція молодих учених та студентів. Збірник наукових праць.– Хмельницький: Гонта А.С., 2014. – С.329-338

References

1. S.McConnell. Code complete - Microsoft Press, 2013 - 896 p.

2. CHAOS Manifesto: Think big, act small – The Standish Group International: CHAOS Knowledge Center, 2013 – 52 p.

3. ISO 9000:2005 Quality management systems – Fundamentals and vocabulary

4. ISO 9001:2008 Quality management systems – Requirements

5. P.Bourque. Guide to the software engineering body of knowledge (SWEBOK). Version 3.0 / Bourque P., R.E.Fairley - A project of the IEEE Computer Society, 2014 – 335 p.

6. A.V.Turchyn. Structura globalnoy katastrofi: Riski vymiraniya chelovechestva v XXI veke / Turchyn A.V. – M.: Izdatelstvo LKI, 2011 – 432 s.

7. Software goes wrong, we all know that, but just how wrong can it go? // [Electronic resource] – Access mode: http://www.datareservoir.co.uk/bugs /

8. 20 famous software disasters // [Electronic resource] – Access mode: http://sandipsandilya.wordpress.com/2011/01/17/20-famous-software-disasters/

9. Explaining settlement fails // [Electronic resource] – Access mode: http://www.ny.frb.org/research/current_issues/ci11-9/ci11-9.html

10. R.Patton. Software testing: 2nd edition / Patton R. — Indianapolis: Sams, 2005. — 408p.

11. N.G.Levenson. Systemic factors in software-related spacecraft accidents / Levenson N.G.// AIAA Space 2001 Conference and Exposition, pp.1-11

12. N.G.Leveson Software challenges in achieving space safety / Levenson N.G.// Journal of the British Interplanetary Society - Vol. 62, July/August 2009, pp. 265-272

13. T.Ishimatsu. Hazard analysis of complex spacecraft using systems-theoretic process analysis / Ishimatsu T., Leveson N.G., Thomas J.P., Fleming C.H., Katahira M., Miyamoto Yu., Ujiie R., Nakao H., Hoshino N. // Journal of Spacecraft and Rockets - Vol. 51, No. 2 (2014), pp. 509-522

14. C.Jones. The economics of software quality / Jones C., Bonsignour O. – Boston: Pearson Education, 2012. – 588 p.

15. E.Yourdon. Death March: The complete software developer's guide to surviving "Mission impossible" projects (2nd edition) / Yourdon E.– Prentice Hall, 200

16. N.Johnson. Software testing: Innovate or die // [Electronic resource] – Access mode: http://www.slideshare.net/johnsonnigel/testing-innovate-or-die-42187102

17. A.Maedche. Software for people: fundamentals, trends and best practices (Management for professionals) / Maedche A., Botzenhardt A., Neer L. – Berlin: Springer-Verlag Berlin Heidelberg, 2012. – 293 p.

18. A.V.Krasiy. Modeluvannya protsesu prognozuvannya kharakteristik programnogo zabezpechennya na osnovi analizu spetsifikatsiy / Krasiy A.V. // Naukoviy jurnal "Computerno-integrovani tekhnologii: osvits, nauka, virobnictvo" - Lutsk: Lutskiy natsinonalniy tekhnichniy universitet, 2014 - s.66-76

19. N.Fenton. Software metrics: A rigorous approach (3rd edition) – CRC Press, 2014 – 597 p.

20. A.Chen. Visual models for software requirements / Chen A., Beatty J. – W.: MS Press, 2012. – 444 p.

21. A.Fatwanto. Software requirements specification analysis using natural language processing technique / Fatwanto A. // International Conference on Quality in Research – 2013, pp.105-110

22. T.Rehman. Analysis of requirement engineering processes, tools/techniques and methodologies / Rehman T., Khan M.N.A., Riaz N. // I.J. Information Technology and Computer Science – 2013, pp.40-48

23. IEEE 830-1998. Recommended practice for software requirements specifications

24. K.Wiegers. Software requirements: 3-rd edition / Wiegers K., Beatty J. – W.: MS Press, 2013. – 640 p.

25. T.O.Hovorushchenko. Matematichne modeluvannya spetsifikatsii vimog ta kharakteristik programnogo zabezpechennya / Hovorushchenko T.O., Krasiy A.V. // Radioelektronni i computerni systemy– Kharkiv: NAU "KhAI", 2014 – № 5, s.34-39

26. A.V.Krasiy. Neyromerezhna model prognozuvannya kharakteristik [programnogo zabezpechennya na osnovi analizu spetsifikatsiy / Krasiy A.V // Intelektualni tekhnologii v systemnomu programuvanni. III Vseukrainska naukovo-praktichna konferentsiya molodykh uchenykh ta studentiv. Zbirnyk naukovyh prats - Khmelnytskiy: Gonta A.S., 2014 - s.329-338

Рецензія/Peer review : 21.4.2015 р. Надрукована/Printed : 13.5.2015 р.
Рецензент: д.т.н. О.В. Поморова