УДК 004.891.3: 004.3

ПРОЦЕС ПОВТОРНОГО ТЕСТУВАННЯ ПІД ЧАС ЕКСПЕРТИЗИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Т.О. ГОВОРУЩЕНКО

Описано задачі експертизи програмного забезпечення (ПЗ). Запропоновано використовувати повторне тестування з метою перевірки ПЗ відповідно до встановлених вимог та оцінки якості й безпомилковості розробленого ПЗ. Розроблено нейромережну категорійну модель і нейромережний метод процесу повторного тестування ПЗ. Виконано програмну реалізацію та дослідження імітаційної моделі штучної нейронної мережі в пакеті Matlab. Зроблено оцінку підвищення достовірності процесу повторного тестування ПЗ. Розроблено структуру та виконано програмну реалізацію системи визначення необхідності повторного тестування на основі звіту про основне тестування ПЗ.

ВСТУП

Процес розробки програмного продукту подібний до інших інженерних задач. Його найбільш вдала адаптація для програмістів називається уніфікований процес розробки програмного забезпечення (Unified Software Development Process — USDP). Інститутом IEEE було розроблено стандарти документації такого процесу розробки програмного забезпечення (ПЗ). Згідно із цими стандартами, документація проекту складається з таких складових, які розроблялися в порядку перерахування [1]:

- План експертизи програмного забезпечення (Software Verification and Validation Plan — SVVP) описує, яким чином і в якій послідовності мають перевірятися стадії проекту і сам продукт відповідно до вимог.
- План контролю якості ПЗ (Software Quality Assurance Plan — SQAP) показує, яким чином проект має досягти відповідності встановленому рівню якості.
- План управління конфігураціями ПЗ (Software Configuration Management Plan — SCMP) описує, де і як розробник зберігатиме версії документації, програмного коду і як визначатиме, яким чином код пов'язано із документацією.
- План управління програмним проектом (Software Project Management Plan — SPMP) описує, як розробник керуватиме проектом створення програмного продукту, щоб довести розробку до вдалого фіналу.
- Специфікація вимог до програмного забезпечення (Software Requirements Specification — SRS) — це найважливіший документ. Він

складається із двох частин: перша включає те, що «попросив» замовник, друга — що «зроблять програмісти»;

- Проектна документація ПЗ (Software Design Document — SDD) — це архітектура і деталі проектування додатку.
- Документація по тестуванню ПЗ (Software Test Documentation — STD) — опис того, як має проводитись тестування, хто при цьому присутній, результати тестування тощо.

Експертна оцінка програм (peer review) призначена для ефективного та раннього виявлення дефектів у ПЗ і може бути реалізована за допомогою перегляду вихідних текстів, наскрізного структурного контролю або інших методів колективного вивчення.

Основу методології експертизи ПЗ складає оцінка виконання вимог до ПЗ на різних етапах життєвого циклу. При цьому необхідно оцінювати виконання функційних вимог і вимог до цілісності безпеки [2].

МІСЦЕ ПРОЦЕСУ ПОВТОРНОГО ТЕСТУВАННЯ ПІД ЧАС ЕКСПЕРТИЗИ ПЗ

Розглянемо методи оцінки виконання вимог до ПЗ. Така оцінка може проводитись не лише експертами в процесі ліцензування, але й розробником (під час тестування та верифікації) та замовником ПЗ (під час прийняття ПЗ в експлуатацію). Метою проведення оцінки ПЗ є перевірка його відповідно до встановлених вимог. Не менш важливою метою роботи експерта є здійснення реального впливу на підвищення рівня якості та надійності ПЗ. Для цього всі зауваження та рекомендації експертів мають передаватись розробникам для оперативного усунення виявлених недоліків. У результаті сумісної діяльності розробників і експертів можуть вноситись корективи в проект та знижуватись кількість невиявлених дефектів ПЗ. Методи, що застосовуються, можуть включати проведення додаткових незалежних випробувань [2–5].

Підвищити достовірність тестування (імовірність проведення вірного та вичерпного тестування ПЗ, під час якого не припускались помилки) і, відповідно, якість ПЗ можна не тільки шляхом тестування дефектів на етапах розроблення та налагодження (праці відомих вітчизняних та іноземних учених: В.С. Харченка [5, 6], В.А. Гуляєва [7], В.М. Локазюка [8], В.В. Ліпаєва [9–11], Г. Майєрса [12, 13], С. Канера [14], І. Соммервіла [15], Б. Бейзера [16, 17]), а й шляхом повторного тестування з метою виявлення прихованих помилок у програмах після основного тестування. Це підтверджується тим, що достовірність тестування й якість ПЗ залежать від кількості виявлених помилок, а також і прихованих.

Мета роботи — проведення оцінки замовником ПЗ із метою перевірки його відповідно до встановлених вимог і підвищення рівня якості та надійності ПЗ. Для досягнення цієї мети слід використовувати повторне тестування — тестування з метою виявлення прихованих помилок, яке здійснюється після розроблення та налагодження ПЗ і є окремим технологічним процесом [18, 19].

Повторне тестування здійснюється на етапі вхідного контролю, який виконує замовник, тобто допомагає замовнику оцінити якість тестування

програмного забезечення, яке приймається, і вказує на наявність у ньому прихованих помилок [20, 21]. Повторне тестування може застосовуватись на вимогу замовника, або якщо на його необхідність вказують результати основного тестування (за перевищенням порогових значень кількостей помилок кожного рівня).

Під час використання паралельної каскадної моделі життєвого циклу ПЗ можна показати, яке місце відводиться повторному тестуванню в життєвому циклі ПЗ (рис. 1).

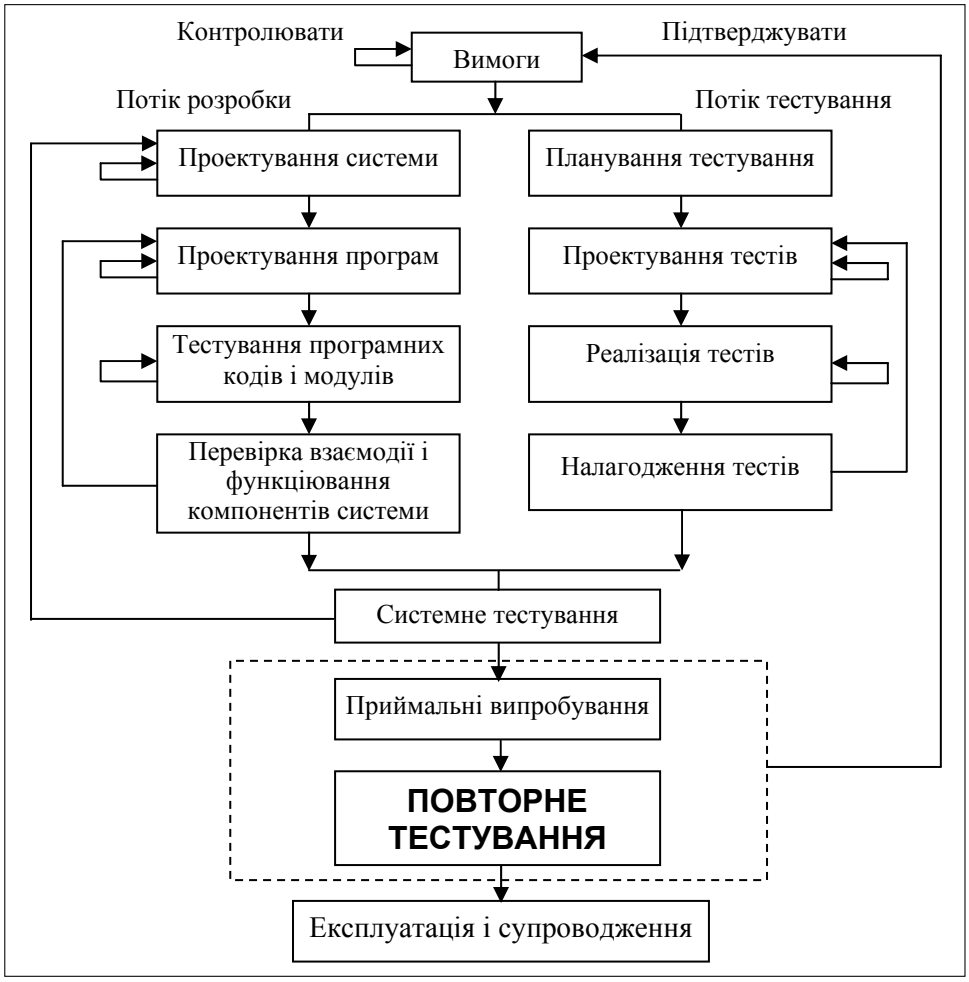

Рис. 1. Місце повторного тестування в паралельній каскадній моделі життєвого циклу програмного забезпечення

НЕЙРОМЕРЕЖНА КАТЕГОРІЙНА МОДЕЛЬ ПРОЦЕСУ ПОВТОРНОГО ТЕСТУВАННЯ ПЗ

Усі приховані помилки розподілимо за їх небезпечністю та ступенем впливу на ПЗ: незначні (НПП), помірні (ППП), серйозні (СПП) та катастрофічні (КПП) приховані помилки. НПП програмного забезпечення вважатимемо такі, що не впливають на дії користувача, програмний продукт із їх наявніс-

тю придатний для використання. ППП програмного забезпечення вважатимемо такі, що впливають на дії користувача. Програмний продукт за їх наявністю придатний для використання з частковою втратою функційності. СПП програмного забезпечення вважатимемо такі, що призводять до помилкових результатів, внаслідок чого програмний продукт непридатний для використання. КПП програмного забезпечення вважатимемо такі, що призводять до спотворення інформації (даних), внаслідок чого програмний продукт непридатний для використання і спроби його опрацювати призводить до відмови програмної системи. НПП присвоїмо найнижчий рівень категорійності — 1. ППП присвоїмо, відповідно, рівень 2; серйозним — рівень 3. Найвищим рівнем вважатимемо КПП — рівень 4. Таким чином, рівнів прихованих помилок буде чотири.

Вперше пропонується концептуальна модель підвищення достовірності тестування ПЗ із виявленням прихованих помилок різних типів шляхом повторного тестування ПЗ із розподілом прихованих помилок на різні категорії і припущенням, що певна кількість помилок попередньої за серйозністю категорії призводить до появи окремих типів помилок наступної категорії, що забезпечило б вибір та обґрунтування моделі процесу на базі ШНМ.

Вірність такої концепції можна легко проілюструвати. Наприклад, потрібно порівняти між собою два аргументи — a і b: якщо $a > b$, то знайти різницю $a - b$, у протилежному випадку — суму $a + b$. Припустимо, що було внесено помилку логічної умови або помилку в гілках true, false, які належать до серйозних помилок. У результаті дії цієї помилки виникає наступна помилка — програма та її функціювання не відповідає специфікації вимог, яка вже належить до катастрофічних.

На основі запропонованого підходу до розподілу прихованих помилок за категорійністю введемо поріг a_i допустимої кількості помилок і важливості помилок різних типів одного виду, при перевищенні якого необхідно здійснювати повторне тестування з метою виявлення прихованих помилок цього виду. Знаходження більшої кількості прихованих помилок під час повторного тестування підвищить, у свою чергу, достовірність процесу тестування взагалі і, відповідно, якість програмного продукту.

На основі запропонованої концепції повторного тестування і розподілу прихованих помилок за категорійністю з урахуванням порогів допустимої кількості помилок і важливості помилок [18, 20, 21], розроблено математичну модель процесу повторного тестування, в основі якої лежить штучна нейронна мережа (ШНМ), структура якої представлена на рис. 2.

Вибір апарату ШНМ мотивований тим, що штучні нейронні мережі завдяки можливості апроксимацій нелінійних функцій надають можливість враховувати важливість (ваги) кожного типу неприхованих та прихованих помилок, пороги граничної кількості допустимих помилок кожної категорії, взаємний вплив прихованих помилок одних типів на помилки інших типів. Визначення вихідного функціоналу кожного із шарів ШНМ, що відповідають категорійності помилок, дає можливість оцінити сумарний вплив кожної категорії помилок на якість розробленого ПЗ, яке пройшло основне тестування, і зробити висновки щодо необхідності повторного тестування з

тування з метою виявлення й усунення помилок тієї чи іншої категорії. Важкоформалізованою задачею повторного тестування є визначення ваг впливу помилок різних типів однієї категорії [18, 20] на помилки іншої категорії, причиною яких є помилки попередніх категорій [18, 20].

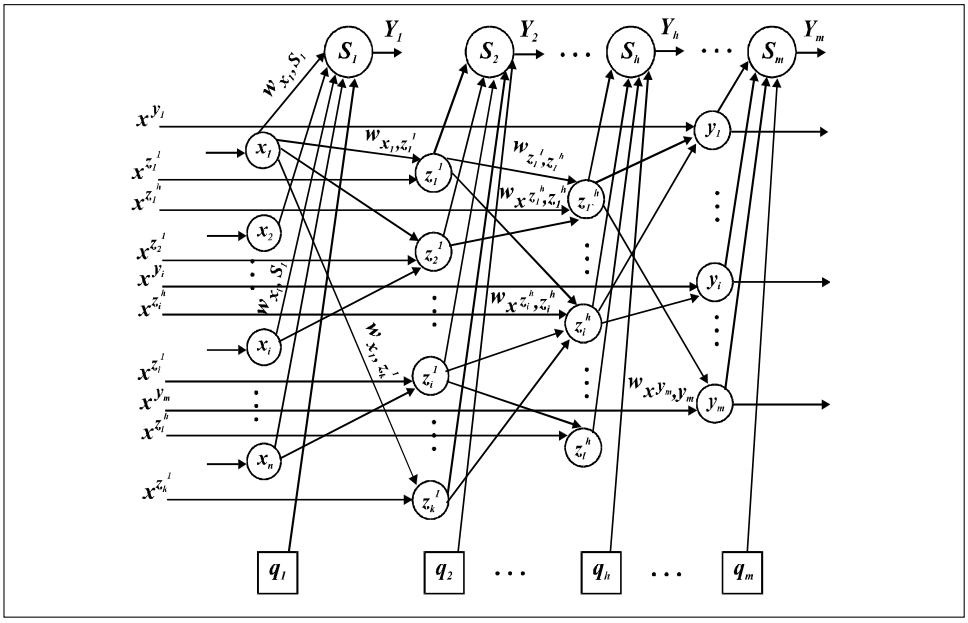

Рис. 2. Нейромережна категорійна модель процесу повторного тестування

НЕЙРОМЕРЕЖНИЙ МЕТОД ПРОЦЕСУ ПОВТОРНОГО ТЕСТУВАННЯ ПЗ

Початковими даними для реалізації повторного тестування є інформація про типи помилок (множина $P = \{p_k \mid k = 1...n\}$), виявлених під час основного тестування, та методи (множина $M = \{m_k \mid k = 1...n\}$) і операції, що були застосовані для їх виявлення (множина $O = \{o_k \mid k = 1...n\}$). Ця інформація береться зі звітів про результати основного тестування. Оскільки основне тестування здійснює тестувальник, то на результати тестування можливий вплив суб'єктивного та людського факторів, що може як позитивно, так і негативно впливати на ефективність повторного тестування. Саме для зменшення зазначеного суб'єктивного фактора враховуються не тільки типи виявлених помилок, а й методи та операції тестування. Суть нейромережного методу процесу повторного тестування ПЗ [22] відображена на рис. 3.

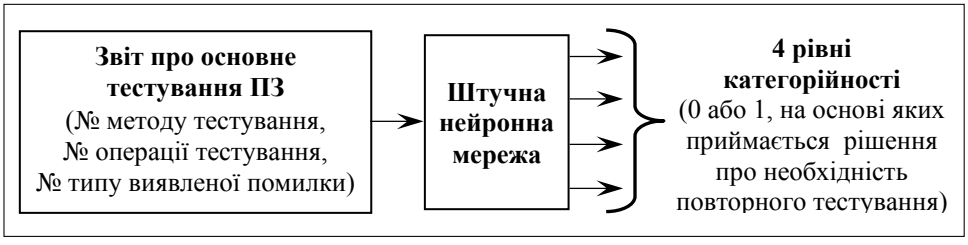

Рис. 3. Принцип застосування ШНМ для процесу повторного тестування

Вхідні дані для реалізації повторного тестування подаються у вигляді матриці $VD = \begin{vmatrix} m_1 & o_1 & p_1 \\ . & . & . \\ m_i & o_i & p_i \\ . & . & . \\ m_n & o_n & p_n \end{vmatrix}$, де m_i, o_i, p_i — елементи множин M, O та P відповідно. Кожен елемент матриці VD, представлений у вигляді тексту, піддається перетворенню для подання його у кількісному вигляді. Використовуючи матриці $MN = \begin{vmatrix} 1 & m_1 \\ . & . \\ i & m_i \\ . & . \\ s & m_s \end{vmatrix}$, де m_i — елемент множини M, $ON = \begin{vmatrix} 1 & o_1 \\ . & . \\ i & o_i \\ . & . \\ v & o_v \end{vmatrix}$, де o_i — елемент множини O, $PN = \begin{vmatrix} 1 & p_1 \\ . & . \\ i & p_i \\ . & . \\ z & p_z \end{vmatrix}$, де p_i — елемент множини P, які представляють собою присвоєння номерів методам тестування, операціям тестування та типам виявлених помилок відповідно, $[i,1]$-й елемент матриці VD, представлений у вигляді тексту, піддається перетворенню для подання його у кількісній формі. Відбувається пошук елемента в другому стовпці матриці MN, одержується порядковий номер j рядка елемента. $[j,1]$-й елемент матриці MN заноситься в $[i,1]$-й елемент матриці $VDM = \begin{vmatrix} mn_1 & on_1 & pn_1 \\ . & . & . \\ mn_i & on_i & pn_i \\ . & . & . \\ mn_n & on_n & pn_n \end{vmatrix}$, де mn_i, on_i, pn_i — кількісне представлення значень елементів множин M, O та P відповідно.

Далі $[i,2]$-й елемент матриці VD піддається перетворенню в кількісне представлення. Відбувається пошук елемента в другому стовпці матриці ON, одержується порядковий номер j рядка елемента. $[j,1]$-й елемент матриці ON заноситься в $[i,2]$-й елемент матриці VDM. Останнім піддається перетворенню в кількісне представлення $[i,3]$-й елемент матриці VD. Відбувається пошук елемента в другому стовпці матриці PN, одержується порядковий номер j рядка елемента. $[j,1]$-й елемент матриці PN заноситься в $[i,3]$-й елемент матриці VDM.

Після одержання кількісного представлення значень кожного елемента матриці VD формується набір вхідних векторів для ШНМ. На вхід q_i подається 1, якщо використовувався відповідний для i-го рівня категорійності

метод основного тестування (дану відповідність наведено в матриці $NMRK = \begin{vmatrix} nm_1 & rk_1 \\ nm_2 & rk_2 \\ nm_3 & rk_3 \\ nm_4 & rk_4 \end{vmatrix}$, де nm — номер методу основного тестування, rk — рівень категорійності). На вхід x'_i подається номер i-ї операції основного тестування on_i ($[i,2]$-й елемент матриці VDM), на вхід x_i подається номер i-го типу виявленої під час основного тестування помилки pn_i ($[i,3]$-й елемент матриці VDM). На всі інші входи подається «0».

ШНМ опрацьовує набір вхідних векторів згідно із методом вирішення задачі повторного тестування та видає матрицю вихідних векторів $VV = \begin{vmatrix} rk_{11} & rk_{12} & rk_{13} & rk_{14} \\ . & . & . & . \\ rk_{i1} & rk_{i2} & rk_{i3} & rk_{i4} \\ . & . & . & . \\ rk_{n1} & rk_{n2} & rk_{n3} & rk_{n4} \end{vmatrix}$, де i-й рядок містить i-й вихідний вектор; елемент rk_{i1} містить значення «нуль» або «одиниця» для рівня категорійності з номером 1 i-го вихідного вектора; елемент rk_{i2} містить значення «нуль» або «одиниця» для рівня категорійності з номером 2 i-го вихідного вектора; елемент rk_{i3} містить значення «нуль» або «одиниця» для рівня категорійності з номером 3 i-го вихідного вектора; rk_{i4} містить значення «нуль» або «одиниця» для рівня категорійності з номером 4 i-го вихідного вектора. Вихідні вектори необхідно піддати перетворенню для одержання результатів у лінгвістичній формі. Для цього використовується матриця присвоєння рівнів категорійності типам прихованих помилок $RK = \begin{vmatrix} 1 & rk_1 \\ 2 & rk_2 \\ 3 & rk_3 \\ 4 & rk_4 \end{vmatrix}$, де rk_i — тип прихованих помилок. Перетворенню з кількісної в лінгвістичну форму піддається окремо кожен вихідний вектор, тобто окремо кожен рядок матриці VV. Для перетворення i-го рядка в ньому відбувається пошук «одиниці», запам'ятовується номер стовпця h та знаходиться $[h,2]$-й елемент матриці RK. Знайдений елемент rk_h є лінгвістичним представленням одержаного результату. Цей елемент заноситься в множину результатів $R = \{rk_k \mid k = 1...n\}$. На основі аналізу складу множини R робиться висновок про необхідність та тип повторного тестування.

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ШНМ У ПАКЕТІ MATLAB

У пакеті Matlab було виконано програмну реалізацію моделі ШНМ. Структурну схему імітаційної моделі ШНМ у пакеті Matlab зображено на рис. 4.

На кожен із входів $q_1 - q_4$ необхідно подати «одиницю», тому що тестування здійснюється одним із методів тестування, які утворюються внаслідок об'єднання двох методів тестування під одним номером, що відображено в матриці присвоєння номерів методам тестування.

За статистикою [24] тестувальник тестує програму не більш як чотирма операціями одного методу тестування, тому на кожен із входів Input2–Input5 можна подати не більше чотирьох номерів операцій тестування. На входи Input2 (x^{Z_1}), Input3 (x^{Z_2}), Input4 (x^{Z_3}), Input5 (x^{Z_4}) подаються номери операцій основного тестування ПЗ.

На вхід Input1 (x) подаються номери результатів операцій основного тестування ПЗ, тобто номери типів помилок виявлених під час основного тестування. Оскільки за статистикою [24] у програмі буває максимум 14–15 помилок, то на даний вхід можна подати не більше 16 типів помилок.

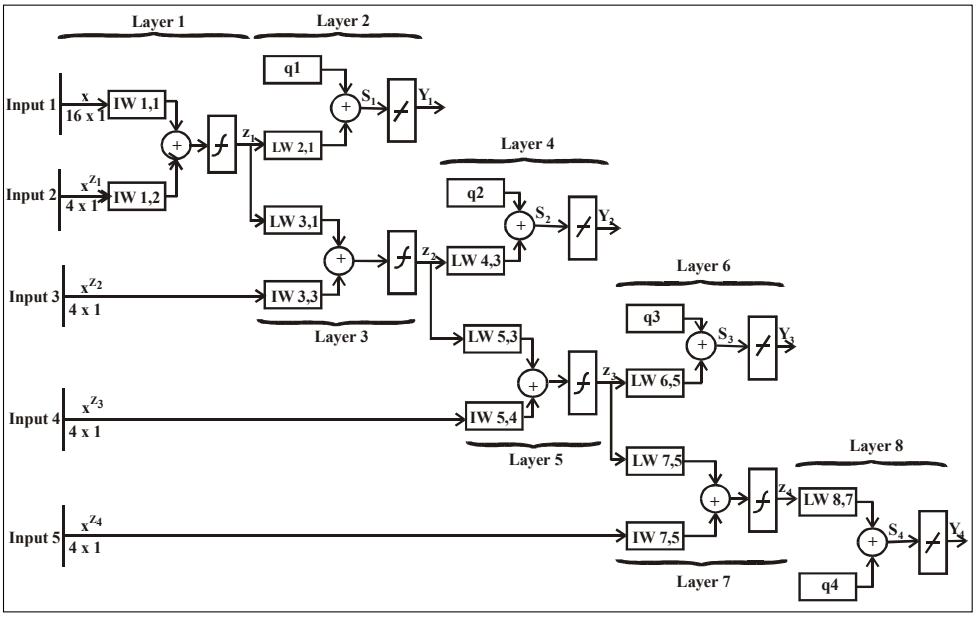

Рис. 4. Структурна схема імітаційної моделі ШНМ у пакеті Matlab

У першому шарі ШНМ (Layer1) значення входів x і x^{Z_1} множимо на відповідні вагові коефіцієнти і додаємо; від одержаної суми беремо активізаційну функцію гіперболічного тангенса, у результаті чого одержимо значення z_1. Це значення передаємо на другий (Layer2) та третій (Layer3) шари ШНМ. У другому шарі до значення z_1, помноженого на відповідний ваговий коефіцієнт, додаємо значення добутку значення q_1 і його вагового коефіцієнта; від одержаної суми S_1 беремо лінійну активізаційну функцію, у результаті чого одержимо значення функції виходу Y_1. У третьому шарі значення z_1, помножене на ваговий коефіцієнт, додається зі значенням добутку значення входу x^{Z_2} і його вагового коефіцієнта; від одержаної суми береться активізаційна функція гіперболічного тангенса, у результаті чого одержуємо значення z_2. Це значення передаємо на четвертий (Layer4) і

п'ятий (Layer5) шари ШНМ, які функціонують аналогічно другому (Layer2) і третьому (Layer3) шарам відповідно. Із четвертого (Layer4) шару одержуємо значення функції виходу Y_2. Із п'ятого шару (Layer5) одержуємо значення z_3, що передається на шостий (Layer6) і сьомий (Layer7) шари ШНМ, які функціонують аналогічно другому (Layer2) і третьому (Layer3) шарам відповідно. Із шостого шару (Layer6) одержуємо значення функції виходу Y_3. Із сьомого шару (Layer7) одержуємо значення z_4, що передається на восьмий шар (Layer8), який функціонує аналогічно другому шару (Layer2). Із восьмого шару одержуємо значення функції виходу Y_4.

Кожен із виходів Y_i відповідає за один із чотирьох рівнів категорійності (Y_1 — перший рівень категорійності, Y_2 — другий рівень категорійності, Y_3 — третій рівень категорійності, Y_4 — четвертий рівень категорійності) і приймає значення «1», якщо штучною нейронною мережею спрогнозовано наявність у програмі помилок i-го рівня категорійності, у протилежному випадку значення виходу Y_i становить «0».

Для вибору алгоритму навчання ШНМ та критерію оцінки якості навчання ШНМ досліджувалась під час навчання вибіркою з 2250 векторів різними алгоритмами із використанням різних критеріїв оцінки якості навчання [20, 23]. У результаті проведеного аналізу було зроблено висновок, що похибка навчання змодельованої ШНМ залежить від критерію оцінки якості навчання та від форми представлення вхідних даних. Тому надалі використовується комбінований критерій якості навчання і масштабована навчальна вибірка. Мінімальна похибка, яку було досягнуто при використанні комбінованого критерію якості навчання та масштабованої навчальної вибірки з 2250 векторів, становить 0,448359. Меншої похибки навчання досягати неможливо і не потрібно, оскільки виходи мережі, які знаходяться в інтервалі [-1; 1] перетворюються для представлення цілими значеннями «1» або «0» (є чи немає помилки *i*-го рівня категорійності відповідно):

$$Y_i = \begin{cases} 1, \text{ якщо } Y_i > 0; \\ 0, \text{ якщо } Y_i \le 0. \end{cases}$$

У результаті аналізу імітаційної моделі ШНМ за часовим показником та за показником «кількість епох» найкращими є: алгоритм навчання CGB на основі методу спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле, алгоритм навчання SCG, алгоритм навчання Флетчера-Рівса, алгоритм навчання Полака-Рібейри та пороговий алгоритм оберненого поширення помилки Rprop. Оскільки алгоритм навчання CGB на основі методу спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле, алгоритм навчання Флетчера-Рівса та алгоритм навчання Полака-Рібейри є модифікаціями методу спряженого градієнта, то для навчання обрано один із них — алгоритм навчання CGB на основі методу спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле. Для тестування ШНМ було побудовано тестову вибірку з 200 векторів, яка також підлягала масштабуванню.

ОЦІНКА ДОСТОВІРНОСТІ ПРОЦЕСУ ПОВТОРНОГО ТЕСТУВАННЯ ПЗ

Із запропонованої моделі випливає, що при $Y_h > 0$ у програмі є помилки категорії, якій відповідає Y_h. Нехай без урахування впливу на початку в програмі було p_x-помилок першого рівня категорійності, p_{z^1}-помилок другого рівня категорійності, ..., p_{z^h}-помилок h-го рівня категорійності. Із урахуванням впливу помилок попереднього рівня категорійності на наступний, помилок стало: p_x — першого рівня категорійності, $p_{z^1} + p_{z_i^{h-1}}$ — h-го рівня категорійності.

За критерій достовірності процесу повторного тестування ПЗ приймемо кількість виявлених помилок згідно із запропонованою моделлю [25, 20].

Достовірність D процесу повторного тестування ПЗ дорівнюватиме:

$$D = kn_1 p_x + kn_2 \frac{p_{z^1} + p_{x_i}}{p_{z^1}} + ... + kn_h \frac{p_{z^h} + p_{z_i^{h-1}}}{p_{z^h}} + ...,$$

де $KN = \{kn_h\}$ — множина коефіцієнтів нормування категорійності прихованих помилок.

Підвищення достовірності процесу повторного тестування ПЗ дорівнюватиме $\Delta D = 1 - \frac{D'}{D}$, де D' — достовірність процесу повторного тестування ПЗ без урахування впливу прихованих помилок кожного попереднього рівня категорійності на помилки наступного рівня категорійності.

Для згаданих раніше чотирьох рівнів категорійності:

$$D = kn_1 p_x + kn_2 \frac{p_{z^1} + p_{x_i}}{p_{z^1}} + kn_3 \frac{p_{z^2} + p_{x_i}^{z^1}}{p_{z^2}} + kn_4 \frac{p_{z^3} + p_{x_i}^{z^2}}{p_{z^3}}.$$

Із j-ї вибірки одержано такі значення величин для визначення достовірності процесу повторного тестування (табл. 1).

Таблиця 1. Значення величин для визначення достовірності повторного тестування

№ експерименту	p_x	p_{z^1}	p_{z^2}	p_{z^3}	p_{x_i}	$p_{x_i}^{z^1}$	$p_{x_i}^{z^2}$
1	28	16	10	4	6	4	2
2	32	20	10	6	8	6	2
3	46	28	14	8	10	6	4
4	50	30	18	10	12	8	2

Експертним шляхом (за результатами роботи дев'ять експертів Хмельницької філії софтверної організації Sitronics Telecom Solutions) присвоєно такі значення коефіцієнтам нормування категорійності прихованих помилок: $kn_1 = 0{,}08$; $kn_2 = 0{,}22$; $kn_3 = 1{,}7$; $kn_4 = 8$.

Підвищення достовірності процесу повторного тестування ПЗ дорівнює:

$$\Delta D_1 = 1 - \frac{12{,}16}{16{,}923} = 0{,}28; \ldots \Delta D_4 = 1 - \frac{13{,}92}{16{,}364} = 0{,}15.$$

Врахування впливу помилок попередніх рівнів категорійності підвищило достовірність процесу повторного тестування ПЗ на 15–28 %.

ПОРОГОВІ ЗНАЧЕННЯ КІЛЬКОСТІ ПОМИЛОК КОЖНОГО РІВНЯ КАТЕГОРІЙНОСТІ

Оскільки уточнення підходу розподілу помилок за пріоритетами і категоріями [24] щодо опису прихованих помилок із введенням концепції категорійності помилок проведено вперше, то порогові значення кількості помилок кожного рівня категорійності, із перевищення яких приймається висновок про необхідність повторного тестування, у відомих літературних джерелах не описано.

Для встановлення цих порогових значень проводились дослідження кількості помилок програмного забезпечення. Програмне забезпечення складалось із різної кількості операторів, із урахуванням і без урахування впливу помилок одного типу (рівня категорійності) на виникнення помилок наступного типу (рівня категорійності) (табл. 2).

Таблиця 2. Кількість помилок програмного забезпечення з урахуванням і без урахування впливу помилок одного типу на виникнення помилок наступного типу

Кількість операторів у програмі	**Кількість виявлених помилок без урахування взаємовпливу помилок**					**Реальна кількість помилок**				
	100	500	1000	5000	10000	100	500	1000	5000	10000
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
Загальна кількість помилок	10	18	25	42	68	16	24	36	42	85
НПП	8	14	19	31	51	8	14	19	31	51
ППП	2	4	5	11	16	5	9	14	11	33
СПП	0	0	1	0	1	2	1	2	0	1
КПП	0	0	0	0	0	1	0	1	0	0

Порогові значення вводяться на основі евристичних оцінок (за табл. 2):

1) якщо кількість помилок 4-го рівня категорійності (катастрофічних) перевищує 1, то повторне тестування необхідне за причини можливості відмови програмної системи;

2) якщо кількість помилок 3-го рівня категорійності (серйозних) перевищує 2, то повторне тестування необхідне за причини виникнення помилок вищого рівня категорійності;

3) якщо кількість помилок 2-го рівня категорійності (помірні) дорівнює або перевищує 50 % від загальної кількості виявлених під час основного

тестування помилок, то повторне тестування необхідне за причини виникнення помилок вищих рівнів категорійності;

4) якщо кількість помилок 1-го рівня категорійності (незначні) дорівнює або перевищує 75 % від загальної кількості виявлених під час основного тестування помилок, то повторне тестування необхідне за причини виникнення помилок вищих рівнів категорійності.

З аналізу табл. 2 видно, що при перевищенні саме таких значень виникають приховані помилки більш високих рівнів категорійності, тому ці значення використовуються як порогові під час формування висновку про необхідність повторного тестування прикладного програмного забезпечення.

ПРОГРАМНІ ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ ПРОЦЕСУ ПОВТОРНОГО ТЕСТУВАННЯ ПЗ

На основі нейромережного методу процесу повторного тестування розроблено структурну схему системи ідентифікації прихованих помилок ПЗ (рис. 5).

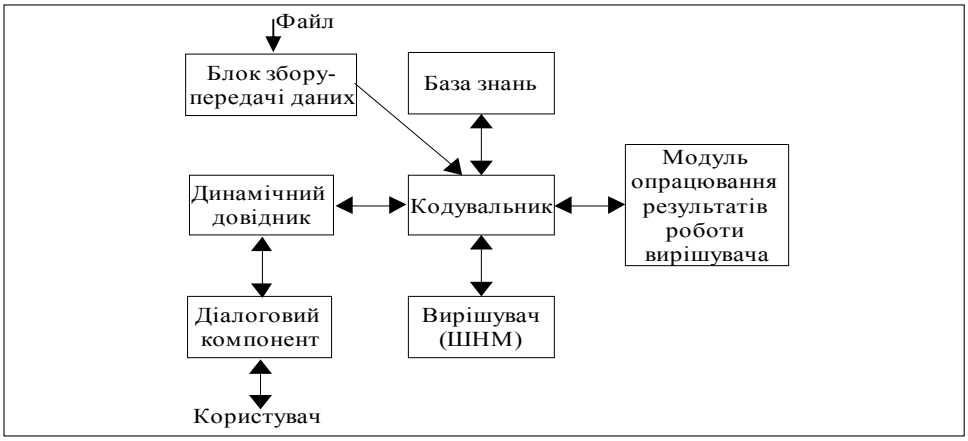

Рис. 5. Структурна схема системи ідентифікації прихованих помилок ПЗ

Система ідентифікації прихованих помилок ПЗ складається з таких компонентів:

1) блок збору – передачі даних — підключає наданий користувачем файл із результатами основного тестування, представленими у вигляді журналу «Метод тестування – Операція тестування – Тип виявленої помилки»;

2) кодувальник — виконує перетворення вхідних даних із лінгвістичної форми представлення в кількісну форму за допомогою відповідних таблиць бази знань та формування вхідних векторів для модуля вирішувача. Здійснює заповнення бази знань вихідними даними, перетворення результуючих векторів вирішувача з кількісної в лінгвістичну форму за допомогою відповідної таблиці бази знань, а також передачу їх модулю опрацювання результатів роботи вирішувача;

3) база знань містить:

- таблиці присвоєння номерів методам і операціям основного тестування, типам виявлених помилок та присвоєння номерів рівням категорійності прихованих помилок;

- таблицю кількісного представлення вхідних даних, в якій містяться вхідні дані, перетворені кодувальником у кількісну форму;
- таблицю текстового представлення результуючих векторів вирішувача (ШНМ), в якій представлені результуючі вектори, перетворені кодувальником в лінгвістичну форму;
- таблиці відповідності методу та операцій основного тестування, типів виявлених під час основного тестування помилок; відповідності між номером методів тестування ПЗ та рівнем категорійності прихованих помилок ПЗ; відповідності між операціями тестування ПЗ та рівнем категорійності прихованих помилок, на основі яких система формує висновок про метод, яким рекомендується проводити повторне тестування прикладного ПЗ;
- правила для формування висновку про необхідність повторного тестування;

4) вирішувач — штучна нейронна мережа, на входи якої подається інформація про методи й операції основного тестування та типи виявлених під час основного тестування помилок, а на виході одержується рівень категорійності прихованих помилок;

5) модуль опрацювання результатів роботи вирішувача — на основі правил та таблиці результатів роботи вирішувача, які взято із бази знань, генерує висновок про необхідність повторного тестування, що передається користувачу через кодувальник, динамічний довідник та діалоговий компонент;

6) динамічний довідник — надає користувачу довідку про формат вхідного файлу, про відомі системі методи і операції основного тестування ПЗ, типи виявлених під час основного тестування помилки ПЗ, а також представляє в наочній формі всі повідомлення будь-якого із компонентів системи;

7) діалоговий компонент — візуалізує повідомлення динамічного довідника та видає їх користувачу в зрозумілій для сприйняття формі.

Запропонована система ідентифікації прихованих помилок програмного забезпечення дозволяє користувачу, на основі звіту про результати основного тестування, одержати висновок про необхідність повторного тестування, а також про наявність у програмному забезпеченні прихованих помилок та їх рівень категорійності.

Систему ідентифікації прихованих помилок ПЗ було реалізовано в Borland C++ Builder 6.0 із застосуванням системи управління базами даних Paradox7.

Під час проведення експериментів розглядались тільки звіти про основне тестування прикладного програмного забезпечення. Наприклад, на вхід системи подавався такий звіт (звіт 1) про результати основного тестування (табл. 3).

Результат роботи системи визначення необхідності повторного тестування ПЗ виводиться в лінгвістичній формі, зрозумілій користувачу, програмісту та тестувальнику. Так, після аналізу звіту 1 із застосуванням введених порогових значень система видає висновок, що повторне тестування не потрібне, оскільки жодне порогове значення не досягнуто.

Таблиця 3. Звіт 1 про результати основного тестування

Метод тестування	Операція тестування	Тип виявленої помилки
Тестування елементів	Перевірка коректності кожної гілки програми (графу керування)	Помилки незалежних маршрутів програми
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка пріоритету арифметичних операцій на правильність і зрозумілість	Помилки обчислень
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності роботи «заглушок»	Некоректна робота «заглушок»
Висхідне тестування, тестування спрягань між елементами	Перевірка правильності розробки та функціювання драйверів	Помилки драйверів та їх розробки
Висхідне тестування, тестування спрягань між елементами	Перевірка даних на втрати при проходженні через спрягання	Помилки спрягань модуля
Висхідне тестування, тестування спрягань між елементами	Перевірка, чи немає несприятливого впливу одного модуля на інший	Помилки спрягань модуля
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка форми операцій	Помилки обчислень
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка логічних операцій на коректність Перевірка пріоритету арифметичних операцій на правильність і зрозумілість	Помилки порівняння Помилки обчислень

ВИСНОВКИ

Експертиза програмного забезпечення полягає в тому, що замовник має переконатись, що розробник виконав поставлені вимоги та розробив якісний безпомилковий продукт із заданою функційністю. Важливою метою роботи експерта в процесі проведення оцінки (експертизи) ПЗ є перевірка ПЗ відповідно до встановлених вимог і здійснення впливу на підвищення якості, достовірності та надійності розробленого ПЗ. Для цього експерти можуть використовувати проведення додаткових незалежних випробувань. Одним із таких випробувань для проведення оцінки замовником ПЗ із метою виявлення прихованих помилок може бути визначення необхідності повторного тестування програмного забезпечення на основі прогнозування наявності прихованих помилок ПЗ та встановлення їх небезпечності та ступеня впливу на ПЗ. Повторне тестування допомагає замовнику оцінити якість програмного забезпечення, яке приймається, та якість тестування цього ПЗ.

Для оцінки небезпечності та ступеня впливу прихованих помилок на ПЗ введено 4 рівні категорійності. На основі запропонованої концепції повторного тестування та розподілу прихованих помилок за рівнями катего-

рійності розроблено нейромережну категорійну модель процесу повторного тестування ПЗ і нейромережний метод процесу повторного тестування ПЗ, дослідження яких дало можливість зробити висновок про підвищення достовірності процесу повторного тестування з врахуванням впливу помилок попередніх рівнів категорійності на 15–28 %. На основі нейромережного методу процесу повторного тестування розроблено структуру та виконано програмну реалізацію системи визначення необхідності повторного тестування, яка на основі звіту про основне тестування ПЗ дає висновок про необхідність повторного тестування ПЗ на основі прогнозу наявності прихованих помилок в аналізованому ПЗ, тобто дає можливість замовнику оцінити якість одержуваного ПЗ.

ЛІТЕРАТУРА

1. *Документирование* программного обеспечения и эффективный процесс разработки. — http://creograf.ru/?messPress_ShowR_161=1.
2. *Скляр В.В.* Оценка качества и экспертиза программного обеспечения: лекционный материал. — Х.: НАУ «ХАИ», 2008. — 204 с.
3. *Ястребенецкий М.А., Васильченко В.Н., Виноградская С.В.* и др. Безопасность атомных станций: Информационные и управляющие системы. — Киев: Техніка, 2004. — 472 с.
4. *Сидельников Ю.В.* Экспертология — новая научная дисциплина // Автоматика и телемеханика. — 2000. — № 2. — С. 107–126.
5. *Харченко В.С., Скляр В.В., Гордеев А.А.* Верификация программного обеспечения. — Х.: НАУ «ХАИ», 2006. — 132 с.
6. *Харченко В.С., Скляр В.В., Тарасюк О.М.* Методы моделирования и оценки качества и надежности программного обеспечения. — Х.: НАУ «ХАИ», 2004. — 159 с.
7. *Гуляев В.А., Коростиль Ю.М.* Диагностирование программного обеспечения микропроцессорных систем. — Киев: Техніка, 1991. — 138 с.
8. *Локазюк В.М.* Надійність, помилки і тестування програмного забезпечення комп'ютерних пристроїв та систем: навч. посіб. — Хмельницький: ТУП, 2003. — 74 с.
9. *Липаев В.В.* Качество программного обеспечения. — М.: Финансы и статистика, 1983. — 263 с.
10. *Липаев В.В.* Отладка сложных программ: Методы, средства, технология. — М.: Энергоатомиздат, 1993. — 384 с.
11. *Липаев В.В.* Тестирование программ. — М.: «Радио и связь», 1986. — 411 с.
12. *Майерс Г.* Надежность программного обеспечения. — М.: Мир, 1980. — 360 с.
13. *Myers G.J.* The Art of Software Testing. — NY: John Wiley and Sons, 1979. — 312 с.
14. *Канер Сэм* и др. Тестирование программного обеспечения: пер. с англ. — Киев: ДиаСофт, 2001. — 544 с.
15. *Соммервил И.* Инженерия программного обеспечения. — 6-е изд. — М.: Изд. дом «Вильямс», 2002. — 624 с.
16. *Beizer B.* Software Testing Techniques. — NY: International Thomson Publishers, 1990. — 503 p.
17. *Beizer B.* Black-Box Testing: Techniques for Functional Testing of Software and Systems. — NY: John Willey & Sons, 1995. — 320 p.
18. *Локазюк В.М., Пантелєєва (Говорущенко) Т.О.* Категорійна модель процесу повторного тестування дефектів програмного забезпечення // Вісн. Технологічного ун-ту Поділля. — 2004. — Т. 1, Ч. 1 — С. 53–58.

19. *Lokazyuk V.M., Govoruschenko T.O.* Category Model of Process of Repeated Software Testing // Proceedings of the Third IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. — Sofia, Bulgaria, 2005. — P. 241–245.
20. *Говорущенко Т.О.* Підвищення достовірності тестування програмного забезпечення // Вісн. Нац. ун-ту «Львівська політехніка» «Комп'ютерні науки та інформаційні технології». — 2007. — С. 186–196.
21. *Lokasyuk V., Pomorova O., Govorushchenko T.* Neural Nets Method for Estimation of the Software Retesting Necessity // Proceedings of the 2008 international workshop on Software Engineering in east and south Europe. — Germany, Leipzig, 2008. — P. 9–14.
22. *Говорущенко Т.О.* Система повторного тестування програмного забезпечення // Радіоелектронні і комп'ютерні системи. — Х.: НАУ «ХАІ», 2005. — № 4. — С. 120–126.
23. *Говорущенко Т.О.* Дослідження моделі вирішувача системи повторного тестування прикладного програмного забезпечення // Вісн. Хмельницького нац. ун-ту. — 2007 — **1**, № 3. — С. 236–244.
24. *Калбертсон Р., Браун К., Кобб Г.* Быстрое тестирование: пер. с англ. — М.: Изд. дом «Вильямс», 2002. — 384 с.
25. *Говорущенко Т.О.* Оцінка ефективності виявлення прихованих помилок у програмному забезпеченні // Вісн. Хмельницького нац. ун-ту. — 2005. — Т. 2, Ч. 1. — С. 190–195.

Надійшла 21.02.2009