УДК 004.891.3: 004.3

ПРОЦЕС ПОВТОРНОГО ТЕСТУВАННЯ В ПРОЦЕСІ ЕКСПЕРТИЗИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Т.О.Говорущенко

Вступ. Процес розробки програмного продукту подібний до інших інженерних задач. Його найбільш вдала адаптація для програмістів називається Уніфікований процес розробки програмного забезпечення (USDP- Unified Software Development Process). Інститутом IEEE було розроблено стандарти документації такого процесу розробки програмного забезпечення (ПЗ). Згідно цих стандартів, документація проекту складається з наступних складових, розроблюваних в порядку перерахування [1]:

- План експертизи програмного забезпечення (SVVP – Software Verification and Validation Plan): документ описує, яким чином і в якій послідовності повинні перевірятись стадії проекту і сам продукт на відповідність вимогам. Іншими словами, цей документ обумовлює те, як замовник і розробник переконаються в тому, що вони зробили те, що хотіли зробити, але не описує, що і як вони робитимуть.
- План контролю якості ПЗ (SQAP – Software Quality Assurance Plan) - показує, яким чином проект повинен досягти відповідності встановленому рівню якості;
- План управління конфігураціями ПЗ (SCMP – Software Configuration Management Plan) - описує, де і як розробник зберігатиме версії документації, програмного коду і як визначатиме, яким чином код пов'язаний з документацією;
- План управління програмним проектом (SPMP – Software Project Management Plan) - описує, як розробник керуватиме проектом створення програмного продукту, щоб довести розробку до вдалого фіналу;
- Специфікація вимог до програмного забезпечення (SRS – Software Requirements Specification) - це найважливіший документ. Складається з двох частин: перша включає те, що "попросив" замовник, друга - те, що "зроблять програмісти";
- Проектна документація ПЗ (SDD – Software Design Document) - це архітектура і деталі проектування додатку;
- Документація по тестуванню ПЗ (STD – Software Test Documentation) - опис того, як повинно проводитись тестування, хто при цьому присутній, результати тестування і т.д.

Експертна оцінка програм (peer review) призначена для ефективного та раннього виявлення дефектів в 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 в процесі експертизи ПЗ. Розглянемо методи оцінки виконання вимог до ПЗ. Така оцінка може проводитись не лише експертами в процесі ліцензування, але й розробником (в ході тестування і верифікації) та замовником ПЗ (при прийнятті ПЗ в експлуатацію). Метою проведення оцінки ПЗ є перевірка його відповідності встановленим вимогам. Не менш важливою метою роботи експерта є здійснення реального впливу на підвищення рівня якості і надійності ПЗ. Для цього всі зауваження та рекомендації експертів повинні передаватись розробникам для оперативного усунення виявлених недоліків. В результаті сумісної діяльності розробників і експертів можуть вноситись корективи в проект і, таким чином, знижуватись кількість невиявлених дефектів ПЗ. Застосовувані методи можуть включати проведення додаткових незалежних випробувань [2-5].

Підвищити достовірність тестування (імовірність проведення вірного і 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 вчених: Харченка В.С. [5, 6], Гуляєва В.А. [7], Локазюка В.М. [8], Ліпаєва В.В. [9 – 11], Майєрса Г. [12, 13], Канера Сема [14], Соммервіла І. [15], Бейзера Б. [16, 17]), а й шляхом повторного тестування з метою виявлення прихованих помилок у програмах після основного тестування. Це підтверджується тим, що, як достовірність тестування, так і якість ПЗ залежать від кількості виявлених у ньому помилок, у тому числі і прихованих.

Для проведення оцінки замовником ПЗ з метою перевірки його відповідності встановленим вимогам та підвищення рівня якості і надійності ПЗ можна 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 здійснює замовник, тобто допомагає замовнику оцінити якість тестування програмного забезечення, яке приймається, і вказує на наявність в ньому прихованих помилок [20, 21]. Повторне тестування може застосовуватись на вимогу замовника або якщо на його необхідність вказують результати основного тестування (по перевищенню порогових значень кількостей помилок кожного рівня).

Використовуючи паралельну каскадну модель життєвого циклу ПЗ, можна показати, яке місце відводиться повторному тестуванню в життєвому циклу програмного забезпечення (рис. 1).

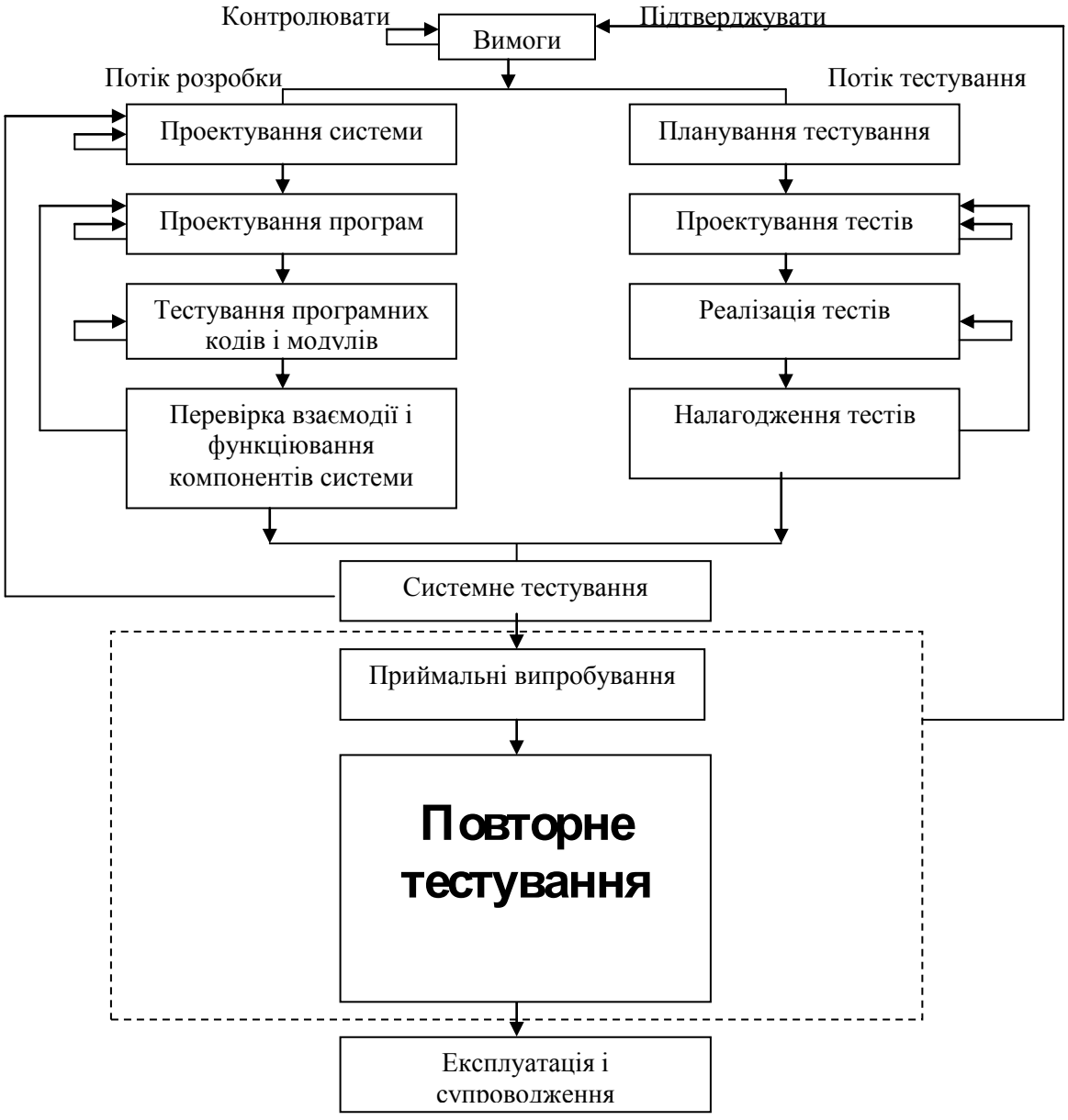

Рис.1. Місце повторного тестування в паралельній каскадній моделі життєвого циклу програмного забезпечення

Нейромережна категорійна модель процесу повторного тестування ПЗ [18-21]. Всі приховані помилки розподілимо за їх небезпечністю та ступенем впливу на ПЗ на незначні (НПП), помірні (ППП), серйозні (СПП) та катастрофічні (КПП) приховані помилки. Незначними прихованими помилками (НПП) програмного забезпечення вважатимемо такі, що не впливають на дії користувача, програмний продукт з їх наявністю придатний для використання. Помірними прихованими помилками (ППП) програмного забезпечення вважатимемо такі, що впливають на дії користувача. Програмний продукт з їх наявністю придатний для використання з частковою втратою функційності. Серйозними прихованими помилками (СПП) програмного забезпечення вважатимемо такі, що призводять до помилкових результатів, внаслідок чого програмний продукт непридатний до використання. Катастрофічними прихованими помилками (КПП) програмного забезпечення

вважатимемо такі, що призводять до спотворення інформації (даних), внаслідок чого програмний продукт непридатний до використання і намагання його опрацювати призводить до відмови програмної системи. Незначним прихованим помилкам присвоїмо найнижчий рівень категорійності – перший. Помірним прихованим помилкам присвоїмо, відповідно, рівень 2; серйозним – рівень 3. Найвищим рівнем вважатимемо катастрофічний – рівень 4. Таким чином, рівнів прихованих помилок буде чотири.

Вперше пропонується концептуальна модель підвищення достовірності тестування ПЗ з виявленням прихованих помилок різних типів шляхом повторного тестування ПЗ з 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 вибір та обгрунтування категорійної моделі процесу повторного тестування на базі ШНМ.

Припустимо, що певна множина незначних помилок (НПП′) призводить до появи підмножини окремих типів помірних помилок (ППП′), які, в свою чергу, призводять до появи певної підмножини серйозних помилок (СПП′), а вони, відповідно, до - катастрофічних помилок (КПП′).

Вірність такої концепції можна проілюструвати наступним простим прикладом.

Приклад 1. Потрібно порівняти між собою 2 аргументи – *a* і *b*: якщо $a > b$, то знайти різницю $a - b$, в протилежному випадку – суму $a + b$.

Вірна блок-схема алгоритму розв'язання такої задачі зображена на рис.2. Припустимо, що було внесено помилку логічної умови (рис.3) або помилку в гілках true, false (рис.4), які належать до серйозних помилок. В результаті дії цієї помилки виникає наступна помилка – програма та її функціювання не відповідає специфікації вимог, яка вже належить до катастрофічних.

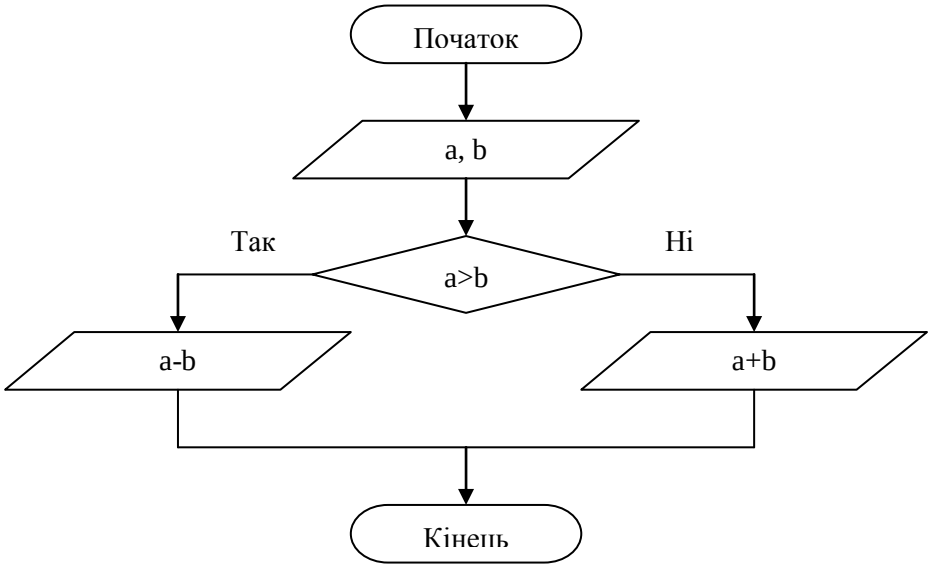

Рис.2. Вірна блок-схема алгоритму розв'язування задачі прикладу 1.

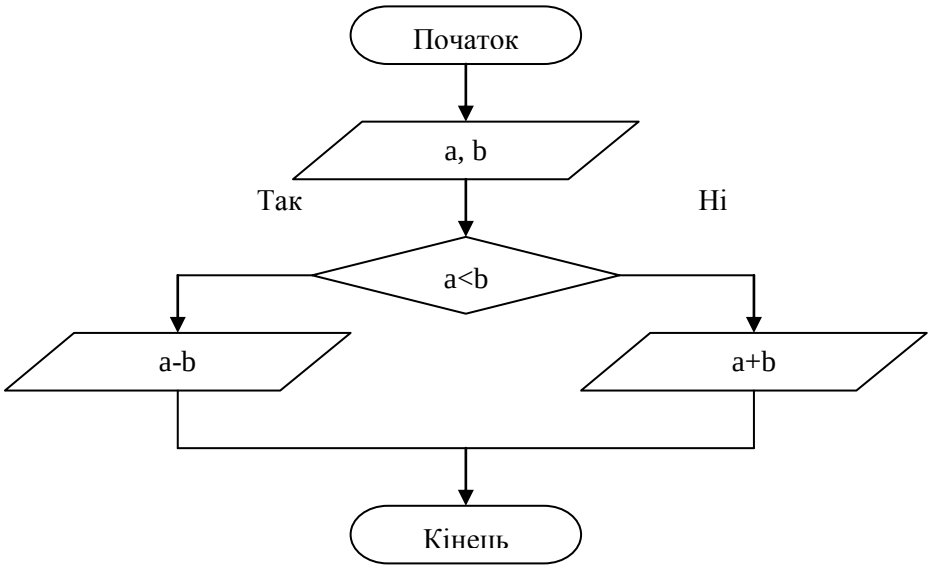

Рис.3. Блок-схема алгоритму розв'язування задачі прикладу 1 з помилкою логічної умови

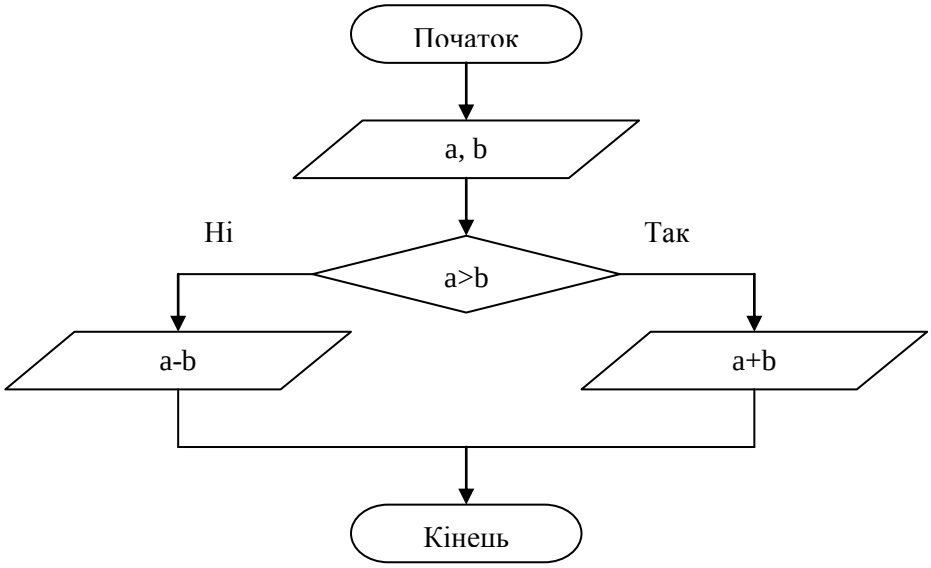

Рис.4. Блок-схема алгоритму розв’язування задачі прикладу 1 з помилкою в гілках true, false

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.

У той же час матимемо на увазі, що помилки одного рівня категорійності можуть бути причиною виникнення помилок не тільки наступного рівня категорійності, але й інших видів помилок вищих рівнів категорійності.

Твердження 1. Якщо сумарна кількість і важливість помилок певного виду не перевищує поріг допустимої кількості помилок і важливості помилок різних типів одного виду Q_i, то причиною наявності помилок інших рівнів категорійності не є помилки зазначеного рівня категорійності.

На основі запропонованої концепції повторного тестування і розподілу прихованих помилок за категорійністю з врахуванням порогів допустимої кількості помилок і важливості помилок [18, 20, 21] розроблено математичну модель процесу повторного тестування, в основі якої лежить штучна нейронна мережа (ШНМ) типу прямонапрямленого персептрону.

Вибір апарату ШНМ мотивований тим, що штучні нейронні мережі за рахунок можливості апроксимацій нелінійних функцій дають 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 метою виявлення і 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]. Ця задача може бути вирішена за допомогою використання навченої ШНМ.

Відобразимо зазначений підхід узагальненою складною ШНМ, в якій структура багатошарового персептрона типу MLP (multi-layer-perceptron) поєднується зі структурою простого персептрона Розенблатта (одношарова ШНМ, де ваги першого шару незмінні, і зважені компоненти вхідного вектора на вході нейрона другого шару сумуються). Другий шар персептрона Розенблатта складається з одного нейрона. Структура цієї ШНМ представлена на рис.5.

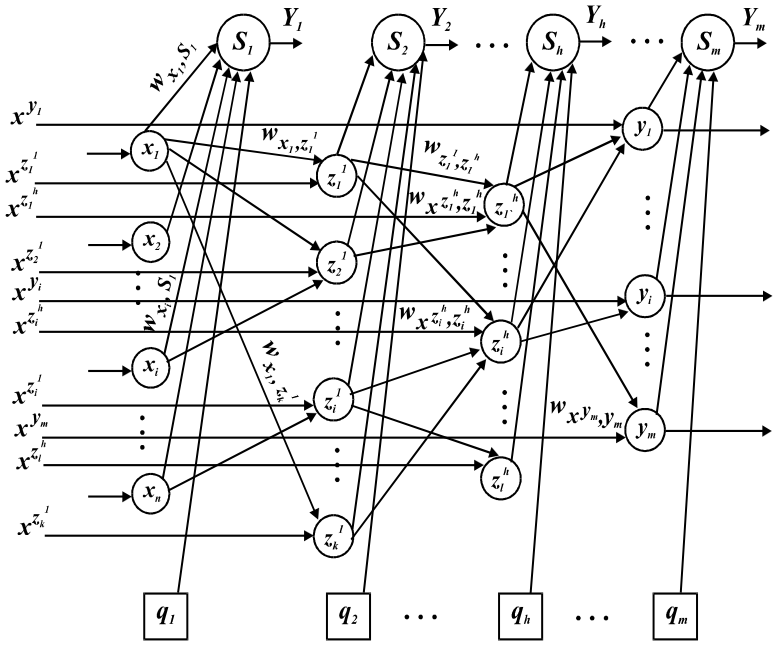

Рис.5. Нейромережна категорійна модель процесу повторного тестування

Нейромережний метод процесу повторного тестування ПЗ [22]. 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 із звітів про результати основного тестування, які надаються тестувальником у вигляді журналу "Метод тестування – Операція, яка виконується під час тестування – Результат операції (тип виявленої помилки)". Оскільки основне тестування здійснює тестувальник, то на результати тестування можливий вплив суб'єктивного та людського факторів (врахування «почерку» тестувальника), що може як позитивно, так і негативно впливати на ефективність повторного тестування. Саме для зменшення зазначеного суб'єктивного фактора враховуються не тільки кількість і типи виявлених помилок, а й методи та операції тестування.

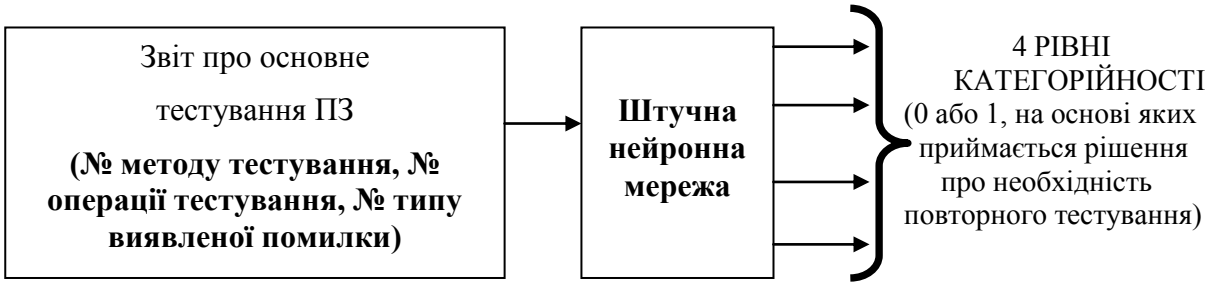

Рис.6. Принцип застосування ШНМ для процесу повторного тестування

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, P відповідно. Кожен елемент матриці VD, представлений у вигляді тексту, піддається перетворенню для представлення 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 представлення його у кількісній формі. Відбувається пошук елемента в другому стовпці матриці MN, одержується порядковий номер j рядка елементу. $[j,1]$-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, P відповідно.

Далі піддається перетворенню в кількісне представлення $[i,2]$-й елемент матриці VD. Відбувається пошук елемента в другому стовпці матриці ON, одержується порядковий номер j рядка елементу. $[j,1]$-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 елементу. $[j,1]$-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 (дана відповідність наведена 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 методу 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 потрібно 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 в пакеті Matlab [20, 23]. В пакеті Matlab було виконано програмну реалізацію моделі ШНМ. Структурну схему імітаційної моделі ШНМ в пакеті Matlab представлено на рис.7.

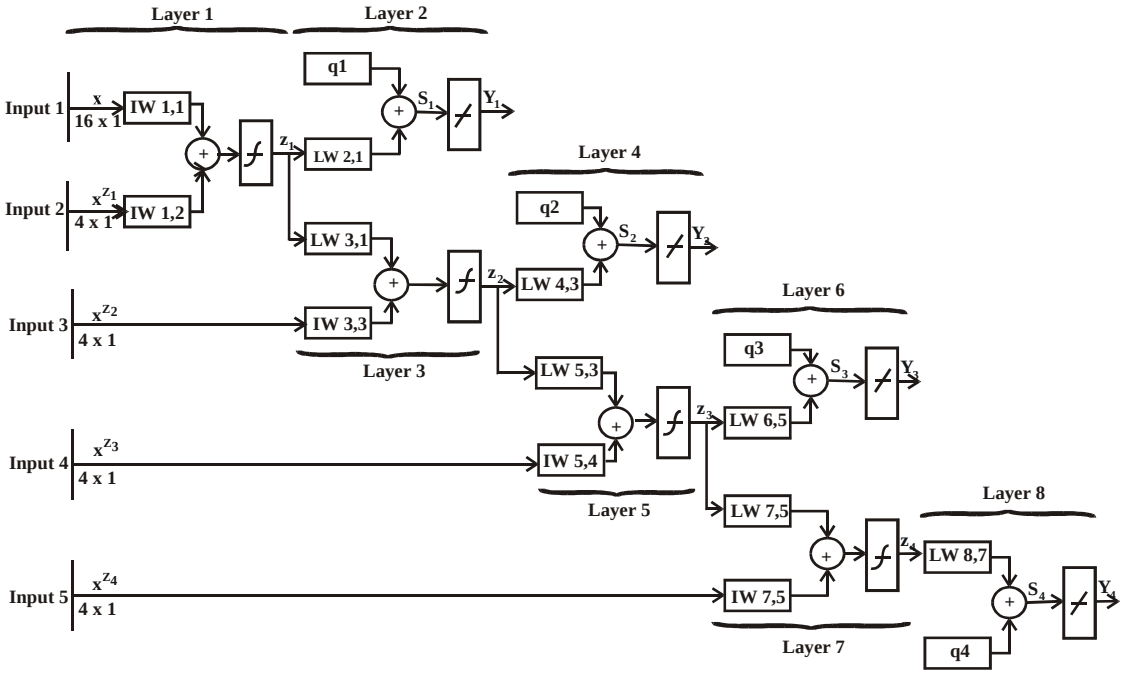

Рис.7. Структурна схема імітаційної моделі ШНМ в пакеті Matlab

На кожен з входів q_1 - q_4 потрібно подати “одиницю”, тому що тестування здійснюється одним з 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 (відповідність між методами тестування та їх операціями наведена в базі знань), тому на кожен з 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. Причому на вхід Input2 подаються номери таких операцій, при виконанні яких найчастіше виявляються помилки першого рівня категорійності (незначні); на вхід Input3 подаються номери таких операцій, виконання яких призводить до виявлення помилок другого рівня категорійності (помірні); на вхід Input4 подаються номери таких операцій, виконання яких дозволяє виявити помилки третього рівня категорійності (серйозні); на вхід Input5 подаються номери таких операцій, які дозволяють виявити помилки четвертого рівня категорійності (катастрофічні). Дана відповідність наведена в базі знань.

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

Значення входів ШНМ лежать в діапазонах, вказаних у таблиці 1 (нумерація методів і операцій тестування ПЗ наведена в матрицях присвоєння номерів методам тестування та операціям тестування).

Таблиця 1 - Діапазони входів ШНМ

Вхід	**Діапазон**
(1)	**(2)**
q_1	0..1
q_2	0..1
q_3	0..1
q_4	0..1
Input1 (x)	0..22

(1)	(2)
Input2 (x^{Z_1})	0, 20..32, 50, 51
Input3 (x^{Z_2})	0, 41..46, 52, 53
Input4 (x^{Z_3})	0, 10..19, 33..40
Input5 (x^{Z_4})	0, 1..9, 47..49

В 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, в результаті чого одержимо значення z_1. Це значення передаємо на другий (Layer2) та третій (Layer3) шари ШНМ. В другому шарі до значення z_1, помноженого на відповідний ваговий коефіцієнт, додаємо значення добутку значення q_1 і його вагового коефцієнта, від одержаної суми S_1 беремо лінійну активізаційну функцію, в результаті чого одержимо значення функції виходу Y_1. В третьому шарі значення z_1, помножене на ваговий коефіцієнт, сумується зі значенням добутку значення входу x^{Z_2} і його вагового коефіцієнта, від одержаної суми береться активізаційна функція гіперболічного тангенса, в 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. З четвертого (Layer4) шару одержуємо значення функції виходу Y_2. З 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. З шостого шару (Layer6) одержуємо значення функції виходу Y_3. З сьомого шару (Layer7) одержуємо значення z_4, що передається на восьмий шар (Layer8), який функціонує аналогічно другому шару (Layer2). З восьмого шару (Layer8) одержуємо значення функції виходу Y_4.

Кожен з виходів Y_i відповідає за один з 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 в програмі помилок i -го рівня категорійності, в протилежному випадку значення виходу Y_i становить „0".

Для вибору алгоритму навчання ШНМ та критерію оцінки якості навчання ШНМ досліджувалась при навчанні вибіркою з 2250 векторів різними алгоритмами з використанням різних критеріїв оцінки якості навчання [20, 23].

В 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):

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

В результаті аналізу імітаційної моделі ШНМ за часовим показником та за показником "кількість епох" найкращими є: алгоритм навчання CGB на основі метода спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле, алгоритм навчання SCG, алгоритм навчання Флетчера-Рівса, алгоритм навчання Полака-Рібейри, пороговий алгоритм оберненого поширення помилки Rprop. Оскільки алгоритм навчання CGB на основі метода спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле, алгоритм навчання Флетчера-Рівса та алгоритм навчання Полака-Рібейри є модифікаціями метода спряженого градієнта, то для навчання оберано один з них – алгоритм навчання CGB на основі метода 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. Процес навчання і тестування алгоритмом навчання CGB на основі метода спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле з використанням комбінованого критерію якості відображається на рис. 8. На рисунку нижня крива відображає графік навчання, а верхня крива відображає графік тестування ШНМ.

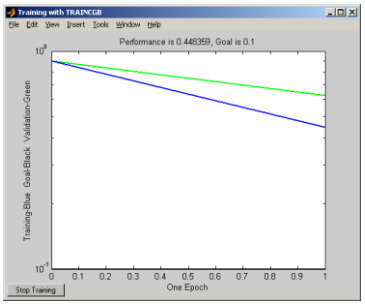

Рис.8. Графіки навчання і тестування ШНМ за алгоритмом навчання CGB на основі метода спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле

Оцінка достовірності процесу повторного тестування ПЗ [25, 20]. Із запропонованої моделі випливає, що при $Y_h > 0$ у програмі є помилки категорії, якій відповідає Y_h. Нехай без врахування впливу на початку у програмі було p_x помилок першого рівня категорійності, p_{z^1} - помилок другого рівня категорійності, ..., p_{z^h} - помилок h-го рівня категорійності. З врахуванням 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 запропонованої моделі.

Достовірність D процесу повторного тестування ПЗ дорівнюватиме:

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

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 врахування впливу прихованих помилок кожного попереднього рівня категорійності на помилки наступного рівня категорійності.

Для згаданих раніше чотирьох рівнів категорійності:

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

З j-ї вибірки одержано наступні значення величин для визначення достовірності процесу повторного тестування (табл. 2).

Таблиця 2 - Значення величин для визначення достовірності повторного тестування

№ експ.	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 9 експертів Хмельницької філії софтверної організації Sitronics Telecom Solutions) присвоєно наступні значення коефіцієнтам нормування категорійності прихованих помилок: $kn_1 = 0{,}08$; $kn_2 = 0{,}22$; $kn_3 = 1{,}7$; $kn_4 = 8$.

Достовірність процесу повторного тестування ПЗ з j-ї вибірки представлена на графіку (рис.9).

Підвищення достовірності процесу повторного тестування ПЗ дорівнює:

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

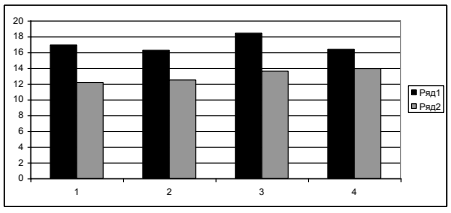

Рис.9. Гістограма підвищення достовірності

На гістограмі ряд 1 (чорний колір) показує достовірність D процесу ідентифікації прихованих помилок при повторному тестуванні, а ряд 2 (сірий колір) - відображає достовірність D' процесу ідентифікації прихованих помилок ПЗ без врахування впливу прихованих помилок кожного попереднього рівня категорійності на помилки наступного рівня категорійності.

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

Порогові значення кількостей помилок кожного рівня категорійності.

Щодо розподілу помилок ПЗ взагалі за видами і впливом на роботу комп'ютерних систем, то в літературі відомий їх розподіл за пріоритетами і категоріями [24]. Оскільки уточнення цього підходу щодо опису прихованих помилок з введенням концепції категорійності помилок проведено вперше, то порогові значення кількостей помилок кожного рівня категорійності, по перевищенню яких приймається висновок про необхідність повторного тестування, в відомих літературних джерелах не описані.

Для встановлення цих порогових значень проводились дослідження кількості помилок програмного забезпечення, яке складалось з різної кількості операторів, з врахуванням і без врахування впливу помилок одного типу (рівня категорійності) на виникнення помилок наступного типу (рівня категорійності). Результати такого дослідження відображені в таблиці 3.

Таблиця 3 - Кількість помилок програмного забезпечення з врахуванням і без врахування впливу помилок одного типу на виникнення помилок наступного типу

	Кількість виявлених помилок без врахування впливу помилок одного типу на виникнення помилок наступного типу					Реальна кількість помилок				
Кількість операторів в програмі	100	500	1000	5000	10000	100	500	1000	5000	10000
Загальна кількість помилок	10	18	25	42	68	16	24	36	42	85
Незначні помилки (1 РК)	8	14	19	31	51	8	14	19	31	51
Помірні помилки (2 РК)	2	4	5	11	16	5	9	14	11	33
Серйозні помилки (3 РК)	0	0	1	0	1	2	1	2	0	1
Катаст-рофічні помилки (4 РК)	0	0	0	0	0	1	0	1	0	0

В результаті аналізу дослідження таблиці 3 можна зробити наступні висновки:

1) для програмного коду зі 100 операторів кількість незначних помилок становить 80% (перевищує 75%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникли 3 помірні помилки; кількість помірних помилок складає 50% загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникли 2 серйозні помилки, які спричинили появу катастрофічної помилки;

2) для програмного коду з 500 операторів кількість незначних помилок становить 78% (перевищує 75%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникло 5 помірних помилок; кількість помірних помилок складає 50% загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникла 1 серйозна помилка, яка не спричинила появу катастрофічних помилок;

3) для програмного коду з 1000 операторів кількість незначних помилок становить 76% (перевищує 75%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникло 9 помірних помилок; кількість помірних помилок складає 56% (перевищує 50%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникла 1 серйозна помилка, 2 серйозні помилки спричинили появу катастрофічної помилки;

4) для програмного коду з 5000 операторів кількість незначних помилок становить 74% (не перевищує 75%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу, тому помірних помилок з причини накопичення незначних помилок не виникло; кількість помірних помилок складає 26% загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу, тому серйозних помилок з причини накопичення помірних помилок не виникло, а, отже, не виникли й катастрофічні помилки;

5) для програмного коду з 10000 операторів кількість незначних помилок становить 75% загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникло 17 помірних помилок; кількість помірних помилок складає 49% (не перевищує 50%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу, тому серйозних помилок з причини накопичення помірних помилок не виникло; одна ж серйозна помилка не спричинила появу катастрофічної помилки.

Порогові значення вводяться на основі евристичних оцінок (за таблицею 3) наступним чином:

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.

При аналізі таблиці 3 видно, що при перевищенні саме таких значень виникають приховані помилки більш високих рівнів категорійності, тому ці значення використовуються в якості порогових при формуванні 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 визначення необхідності повторного тестування програмного забезпечення (рис.10).

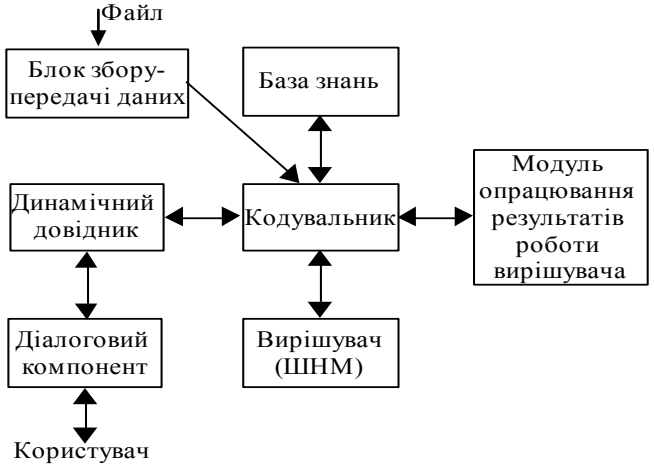

Рис.10. Структурна схема системи визначення необхідності повторного тестування ПЗ

Система визначення необхідності повторного тестування програмного забезпечення складається з наступних компонентів:

1. блок збору – передачі даних – підключає наданий користувачем файл з результатами основного тестування, представленими у вигляді журналу “Метод тестування – Операція тестування – Тип виявленої помилки”;

2. кодувальник – виконує перетворення вхідних даних з лінгвістичної форми представлення в кількісну форму за допомогою відповідних таблиць бази знань та формування вхідних векторів для модуля вирішувача. Кодувальник також перевіряє вхідні дані на правильність та достатність при формуванні вхідних векторів; якщо дані недостатні або вони невірні, то кодувальник передає динамічному довідникові своє повідомлення з пропозицією сформувати ще один файл такої ж форми, як і попередній, з додатковими результатами, що перетворюються в кількісну форму аналогічно даним основного файлу, після чого заносяться в базу знань. Здійснюється заповнення бази знань вихідними даними, перетворення результуючих векторів вирішувача з кількісної в лінгвістичну форму за допомогою відповідної таблиці бази знань та 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 в кількісну форму; таблицю текстового представлення результуючих векторів вирішувача (ШНМ), в якій представлені результуючі вектори, перетворені кодувальником в лінгвістичну форму; таблиці відповідності методу основного тестування, 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, а також правила для формування висновку про необхідність повторного тестування;

4. вирішувач – штучна нейронна мережа, на входи якої подається інформація про методи і 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, взятих з бази знань, генерує висновок про необхідність повторного тестування, який 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 наглядній формі всі повідомлення будь-якого з 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. В даній системі кодувальник виступає в якості інтерфейсу між користувачем і системою, а динамічний довідник – у якості інтерфейсу між користувачем та кодувальником.

Систему ідентифікації прихованих помилок програмного забезпечення 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 наступний звіт (звіт 1) про результати основного тестування (таблиця 4).

Таблиця 4 - Звіт про результати основного тестування

Метод тестування	**Операція тестування**	**Тип виявленої помилки**
(1)	**(2)**	**(3)**
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

(1)	(2)	(3)
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 коректінсть Перевірка пріоритету арифметичних операцій на правильність і зрозумілість	Помилки порівняння Помилки обчислень

Звіт подається в систему визначення необхідності повторного тестування програмного забезпечення по одному рядку (рис.11).

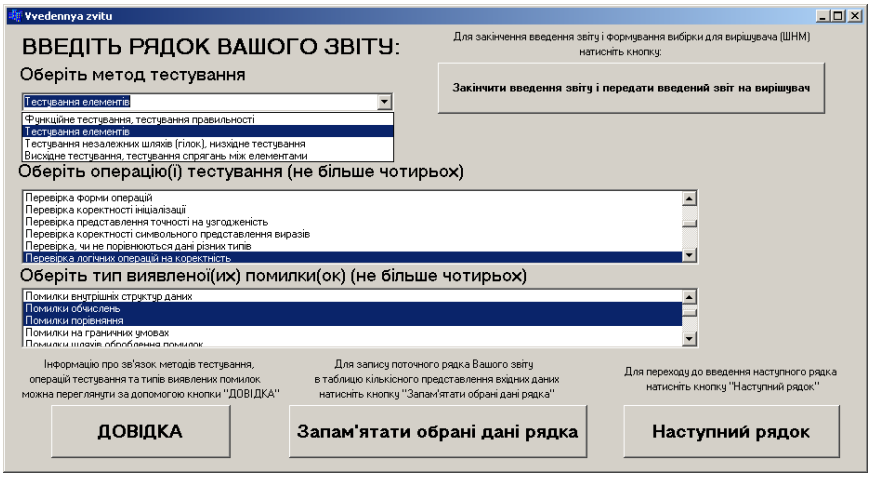

Рис.11. Введення рядка звіту

Результат роботи системи визначення необхідності повторного тестування ПЗ виводиться у лінгвістичній формі, зрозумілій користувачу, програмісту і тестувальнику. Так, після аналізу звіту 1 із застосуванням введених порогових значень система видає висновок, що повторне тестування не потрібне, оскільки жодне порогове значення не досягнуте (рис.12).

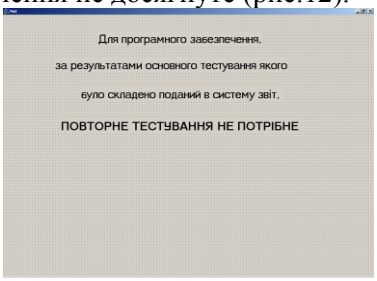

Рис.12. Висновок системи визначення необхідності повторного тестування ПЗ після опрацювання одного із звітів основного тестування ПЗ

Подамо на вхід системи наступний звіт - звіт 2.

Таблиця 5 - Звіт 2 про результати основного тестування

Метод тестування	Операція тестування	Тип виявленої помилки
(1)	(2)	(3)
Функційне тестування, тестування правильності	Перевірка співпадання вихідних результатів з наперед відомими еталонними результатами	Некоректні чи відсутні функції
Функційне тестування, тестування правильності	Перевірка, чи виконує ПС поставлені вимоги	Програма та її функціювання не відповідає специфікації вимог до ПЗ

(1)	(2)	(3)
Тестування елементів	Перевірка коректності кожної гілки програми (графу керування)	Помилки незалежних маршрутів програми
Тестування елементів	Перевірка булевих операторів на коректність, відсутність чи надлишковість	Помилки логічних умов
Тестування елементів	Перевірка цілісності збережуваних даних	Помилки внутрішніх структур даних
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка пріоритету арифметичних операцій на правильність і зрозумілість	Помилки обчислень
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка, чи не порівнюються дані різних типів	Помилки порівняння
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності роботи “заглушок”	Некоректна робота “заглушок”
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності роботи в ситуаціях, коли на верхніх рівнях необхідні результати з нижніх рівнів	Помилки в ситуаціях, коли на верхніх рівнях необхідні результати з нижніх рівнів, які ще не розроблені
Висхідне тестування, тестування спрягань між елементами	Перевірка правильності розробки та функціювання драйверів	Помилки драйверів та їх розробки
Висхідне тестування, тестування спрягань між елементами	Перевірка коректності об’єднання кластерів в загальну структуру	Помилки об’єднання кластерів в загальну структуру
Висхідне тестування, тестування спрягань між елементами	Перевірка, чи не відбувається перехід окремих допустимих неточностей за допустиму межу при інтеграції	Помилки спрягань модуля

Після аналізу звіту 2 система видає висновок, що повторне тестування потрібне - висновок про необхідність повторного тестування зроблено по перевищенню порогового значення кількістю помилок 3-го та 4-го рівня категорійності.

Висновки. Експертиза програмного забезпечення полягає в тому, що замовник повинен переконатись, що розробник виконав поставлені вимоги і розробив якісний безпомилковий продукт із заданою функційністю. Важливою метою роботи експерта в процесі проведення оцінки (експертизи) ПЗ є перевірка відповідності ПЗ встановленим вимогам та здійснення впливу на підвищення якості, достовірності і надійності розробленого ПЗ. Для цього експерти можуть використовувати проведення додаткових незалежних випробувань.Одним з таких випробувань для проведення оцінки замовником ПЗ з метою виявлення прихованих помилок може бути визначення необіхдності 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 і розподілу прихованих помилок за рівнями категорійності 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%. Виконано програмну реалізацію та дослідження імітаційної моделі ШНМ в пакеті Matlab, які дали можливість зробити висновки про: 1) залежність похибки навчання змодельованої ШНМ від критерію оцінки якості навчання та від форми представлення вхідних даних, 2) необхідність використання комбінованого критерію оцінки якості навчання та масштабованої навчальної вибірки, 3) використання алгоритму навчання CGB на основі методу спряженого градієнта, який є найкращим за часовим показником та за показником кількості епох навчання. 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. http://creograf.ru/?messPress_ShowR_161=1
2. Скляр В.В. Оценка качества и экспертиза программного обеспечения. Лекционный материал. - Харьков: НАУ "ХАИ", 2008. - 204 с.
3. Ястребенецкий М.А., Васильченко В.Н., Виноградская С.В. и др. Безопасность атомных станций: Информационные и управляющие системы. - К.: 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. - Харьков: НАУ "ХАИ", 2006. - 132 с.
6. Харченко В.С., Скляр В.В., Тарасюк О.М. Методы моделирования и оценки качества и надежности программного обеспечения. – Харьков: 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: Навчальний посібник. – 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: Пер. с англ. – М.: Мир, 1980. – 360 с.
13. Myers G.J. The Art of Software Testing. – New York: John Wiley and Sons, 1979. – 312 pp.
14. Канер Сэм и др. Тестирование программного обеспечения: Пер. с англ. / Сэм Канер, Джек Фолк, Енг Кек Нгуен. – К.: Издательство "ДиаСофт", 2001. – 544 с.
15. Соммервил И. Инженерия программного обеспечения, 6-е издание: Пер.с англ. – М.: Издательский дом "Вильямс", 2002. – 624 с.
16. Beizer B. Software Testing Techniques. – New York: International Thomson Publishers, 1990. – 503 pp.
17. Beizer B. Black-Box Testing: Techniques for Functional Testing of Software and Systems. – New York: John Willey & Sons, 1995. – 320 pp.
18. Локазюк В.М., Пантелєєва (Говорущенко) Т.О. Категорійна модель процесу повторного тестування дефектів програмного забезпечення // Вісник Технологічного університету Поділля – Хмельницький: ТУП, 2004. – ч.1, т.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 // Вісник Національного університету "Львівська політехніка" "Комп'ютерні науки та інформаційні технології" – Львів: Видавництво Національного університету "Львівська політехніка", 2007 – с.186-196
21. V.Lokasyuk, O.Pomorova, T.Govorushchenko. 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. – pp. 9-14. ISBN 978-1-60558-076-0. (http://doi.acm.org/10/1145/1370868.1370871)
22. Говорущенко Т.О. Система повторного тестування програмного забезпечення // Радіоелектронні і комп'ютерні системи – Харків: НАУ "ХАІ", 2005. - №4, с.120-126
23. Говорущенко Т.О. Дослідження моделі вирішувача системи повторного тестування прикладного програмного забезпечення // Вісник Хмельницького національного університету – Хмельницький: ХНУ, 2007 - №3, т.1, с.236-244
24. Калбертсон Р., Браун К., Кобб Г. Быстрое тестирование: Пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 384 с.
25. Говорущенко Т.О. Оцінка ефективності виявлення прихованих помилок у програмному забезпеченні // Вісник Хмельницького національного університету – Хмельницький, 2005. – ч.1, т.2, с. 190 – 195