УДК 004.891.3: 004.3

Т.О.Говорущенко
Хмельницький національний університет, м.Хмельницький, вул.Інститутська, 11

ПІДВИЩЕННЯ ДОСТОВІРНОСТІ ПРОЦЕСУ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Метою дослідження є підвищення достовірності програмних продуктів за рахунок ідентифікації прихованих помилок шляхом повторного тестування програмного забезпечення (ПЗ). Для досягнення мети розвинута концептульна модель підвищення достовірності тестування ПЗ, розроблена категорійна модель процесу повторного тестування ПЗ на базі штучних нейронних мереж (ШНМ), метод ідентифікації прихованих помилок ПЗ на основі ШНМ, удосконалено метод оцінки достовірності виявлення прихованих помилок ПЗ.

The goal of research is software product reliability increasing at the expense of hidden mistakes identification by repeated software testing. For goal achievement conceptual model of software testing reliability increasing was developed, category neuronet model of repeated software testing process was received, software hidden mistakes identification neuronet technique was developed, technique of evaluation of software hidden mistakes identification reliability was improved

Вступ

Надійність і достовірність роботи сучасних мікропроцесорних пристроїв, комп'ютерів та комп'ютерних систем є запорукою успішного вирішення задач, що розв'язуються ними, та довготривалої їх експлуатації. Одним з основних засобів, за допомогою котрого досягається висока надійність правильного функціювання обчислювальної техніки та вірність розв'язку прикладних задач, є діагностування програмного забезпечення (ПЗ) на різних етапах його життєвого циклу, зокрема, на етапах проектування, розробки та експлуатації.

Тема діагностування програмного забезпечення одержала свій розвиток в працях відомих вітчизняних та іноземних вчених: Харченка В.С., Гуляєва В.А., Локазюка В.М., Савченка Ю.Г., Согомоняна Е.С., Романкевича О.М., Кривулі Г.Ф., Дрозда О.В., Ліпаєва В.В., Майєрса Г., Канера Сема, Соммервіла І., Бейзера Б.

Проведення аналізу програмного забезпечення як об'єкта діагностування дало можливість зробити наступні висновки:

1) розвиток і впровадження нових архітектур та апаратна складність комп'ютерів і систем, побудованих на їх базі, на сьогодні випереджає розроблення методів і засобів діагностування системного та прикладного програмного забезпечення цих систем;

2) існуючі діагностичні програми не завжди повністю враховують зростаючі вимоги до розроблення програм, за причини постійного ускладнення ПЗ та намагання користувачів розв'язувати за його допомогою важкоформалізовані та неформалізовані задачі;

3) низька якість окремих діагностичних програм знижує ефективність використання існуючого ПЗ комп'ютерних систем.

Однією з основних складових діагностування ПЗ є тестування як процес виявлення помилок у програмах. Його роль тим більше зростає в зв'язку з тим, що ПЗ сучасних комп'ютерних систем є досить складним і не може бути бездефектним. Причиною невиявлення помилок у розроблюваному ПЗ швидше за все слід вважати недосконалість тестів, а не бездефектність програми.

З аналізу методів тестування ПЗ стає зрозумілим, що жоден з них не є універсальним і має певні недоліки.

Виявлення недоліків методів та засобів тестування ПЗ, у тому числі ідентифікація прихованих помилок програмних продуктів, є актуальною задачею розвитку методів тестування ПЗ, що підвищує його достовірність.

Концепція повторного тестування програмного забезпечення

Прихованою помилкою назвемо будь-яку помилку ПЗ, що залишилась у програмному продукті після його діагностування у процесі розроблення та налагодження.

Основним тестуванням вважатимемо тестування дефектів (помилок), яке здійснюється на етапах розроблення та налагодження ПЗ і є частковою складовою цих процесів.

Повторним тестуванням вважатимемо тестування з метою виявлення прихованих помилок, яке здійснюється після розроблення та налагодження ПЗ і є окремим технологічним процесом.

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. Повторне тестування в паралельній каскадній моделі [1] представлено на рис.1.

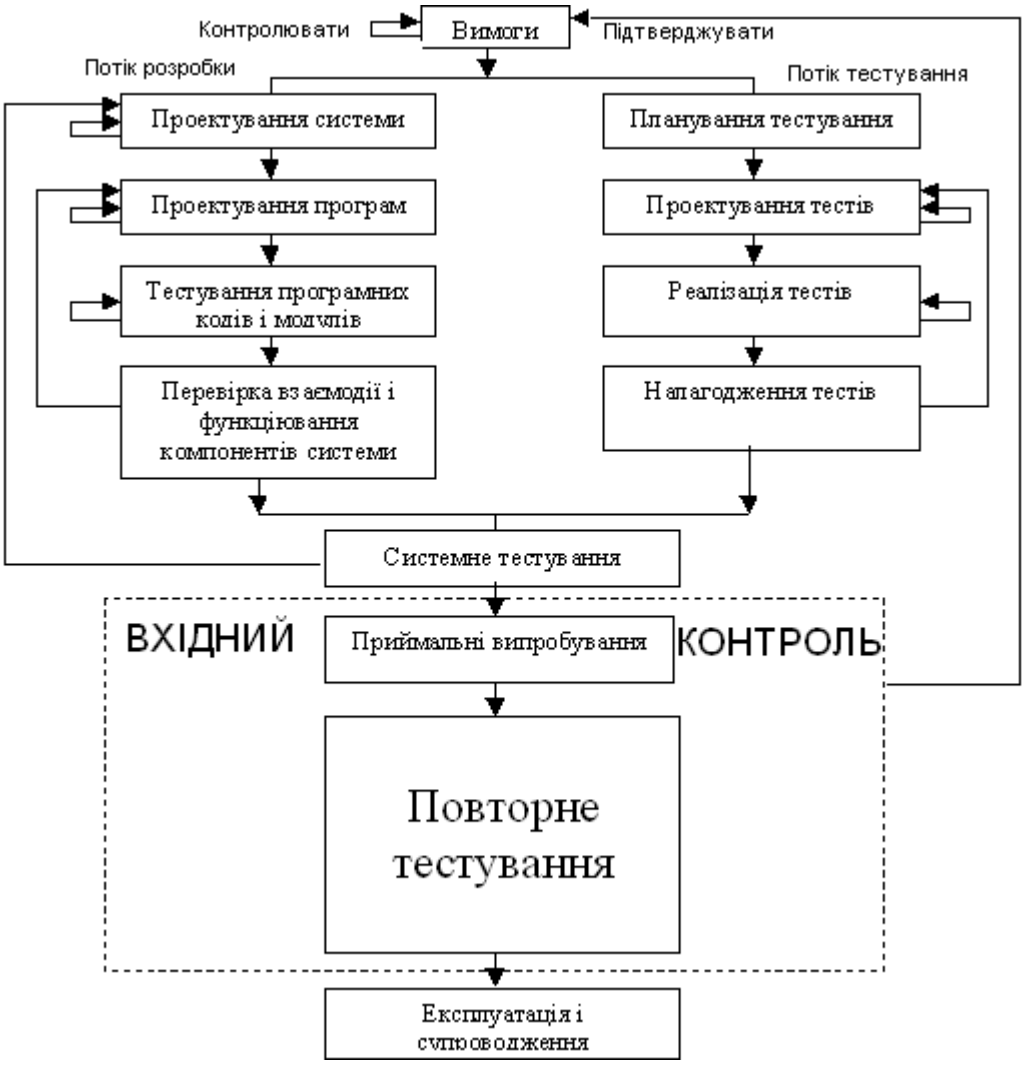

Рис.1. Повторне тестування в паралельній каскадній моделі життєвого циклу програмного забезпечення

Відомий розподіл помилок за пріоритетами і категоріями [1]. Проведемо уточнення розподілу помилок ПЗ щодо опису прихованих помилок [2, 3]. Всі приховані помилки розподілимо за видами на незначні (НПП), помірні (ППП), серйозні (СПП) та катастрофічні (КПП).

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.

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′**).

Введемо поріг $\mathbf{a_i}$ допустимої кількості помилок і важливості помилок різних типів одного виду, при перевищенні якого необхідно здійснювати повторне тестування з метою виявлення прихованих помилок цього виду.

Категорійна модель процесу повторного тестування

На основі запропонованої концепції підвищення достовірності тестування розробимо математичну модель процесу повторного тестування, поклавши в її основу штучну нейронну мережу (ШНМ) типу прямонапрямленого персептрону [2, 3].

Вибір апарату ШНМ мотивований тим, що штучні нейронні мережі 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.

Задачею повторного тестування є визначення ваг впливу помилок різних типів однієї категорії на помилки іншої категорії, причиною яких є помилки попередніх категорій. Ця задача може бути вирішена методом навчання ШНМ.

Відобразимо зазначений підхід узагальненою складною ШНМ, в якій структура багатошарового персептрона типу MLP (multi-layer-perceptron) поєднується зі структурою простого персептрона Розенблатта. Структура цієї ШНМ представлена на рис.2.

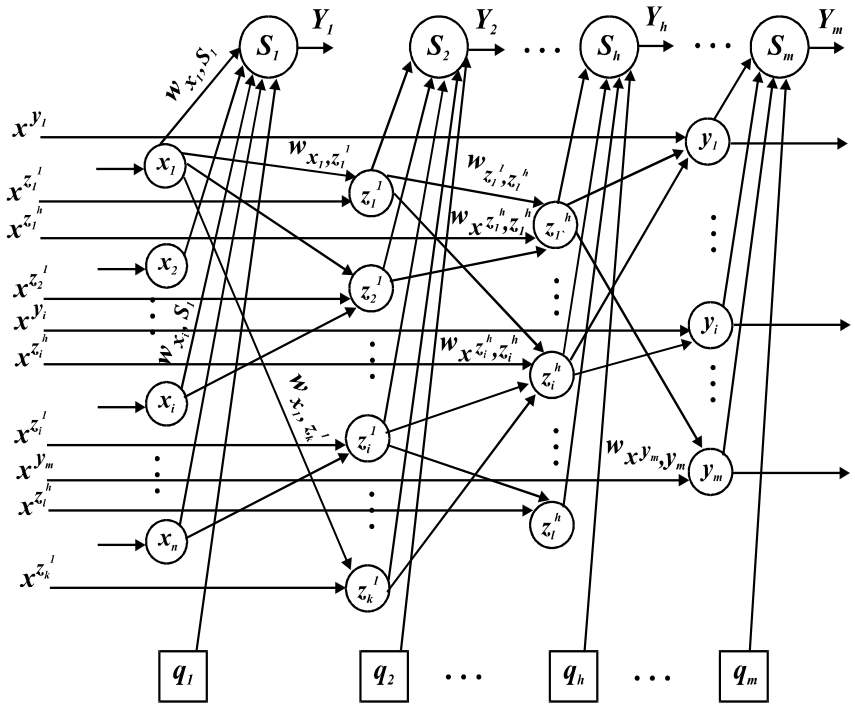

Рис.2. Категорійна модель на базі ШНМ, що відображає зв’язок помилок ПЗ різних категорій

Функціонал ШНМ Y_1 визначиться за формулою:

$$Y_1 = F_{S_1}\left(\sum_{i=1}^{n} x_i w_{x_i} - q_1 w_{q_1}\right).$$

Визначимо функціонал ШНМ Y_h:

$$Y_h = F_{S_h}\left(\sum_{i=1}^{h}\left(f(z^h)\cdot\left(\sum_{i=1}^{h} z_i^{h-1} w_{z_i^{h-1},z_i^h} + \sum_{i=1}^{\ell} x^{z_i^h} w_{x^{z_i^h},z_i^h}\right)\right)\cdot w_{z_i^h,S_h}\right) - q_h w_{q_h,S_h}$$

де $f(z^h)$ - активаційна функція нейронів прихованого шару h; z_i^{h-1} - вихідне значення активності i-го нейрона прихованого шару $(h-1)$; $w_{z_i^{h-1},z_i^h}$ - ваговий коефіцієнт зв'язку між i-ми нейронами прихованого шару ШНМ $(h-1)$ і прихованого шару h; $x^{z_i^h}$ - i-й вхід ШНМ, який безпосередньо пов'язаний з i-м нейроном шару h; $w_{x^{z_i^h},z_i^h}$ - ваговий коефіцієнт зв'язку між входом $x^{z_i^h}$ і i-м нейроном шару h; $w_{z_i^h,S_h}$ - ваговий коефіцієнт зв'язку між i-м нейроном шару h і нейроном S_h.

Активізаційною функцією нейронів прихованих (асоціативних) шарів є функція гіперболічного тангенсу. Активізаційною функцією нейронів ефекторних шарів є лінійна функція. Результати лінійної активізаційної функції нейронів ефекторних шарів лежать в інтервалі [-1; 1]. Нас цікавлять результати у вигляді 1 або 0 (відповідно є помилка *i*-го рівня категорійності чи немає), тому здійснюється перетворення (заокруглення) результатів наступним чином:

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

Оцінка достовірності ідентифікації прихованих помилок програмного забезпечення

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 ідентифікації помилок ПЗ приймемо кількість виявлених помилок згідно запропонованої моделі.

Достовірність D процесу ідентифікації прихованих помилок ПЗ шляхом повторного тестування 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 ідентифікації прихованих помилок дорівнюватиме $\Delta D = 1 - \frac{D'}{D}$, де D' - достовірність процесу ідентифікації прихованих помилок 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. 1).

Таблиця 1

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 присвоїмо наступні значення коефіцієнтам нормування категорійності прихованих помилок: $kn_1 = 0{,}08$; $kn_2 = 0{,}22$; $kn_3 = 1{,}7$; $kn_4 = 8$.

Достовірність процесу виявлення прихованих помилок при повторному тестуванні з j-ї вибірки представлена на графіку (рис.3).

Підвищення достовірності процесу ідентифікації прихованих помилок дорівнюватиме:

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

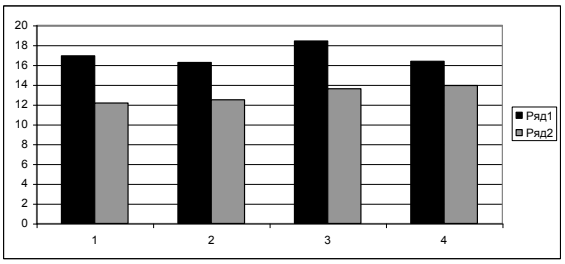

Рис.3. Гістограма підвищення достовірності

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 виявлення прихованих помилок на 15-28%.

Модель та програмна реалізація ШНМ в пакеті Matlab.

В пакеті Matlab було виконано програмну реалізацію моделі ШНМ [4, 5]. Структурну схему імітаційної моделі ШНМ в пакеті Matlab представлено на рис.4.

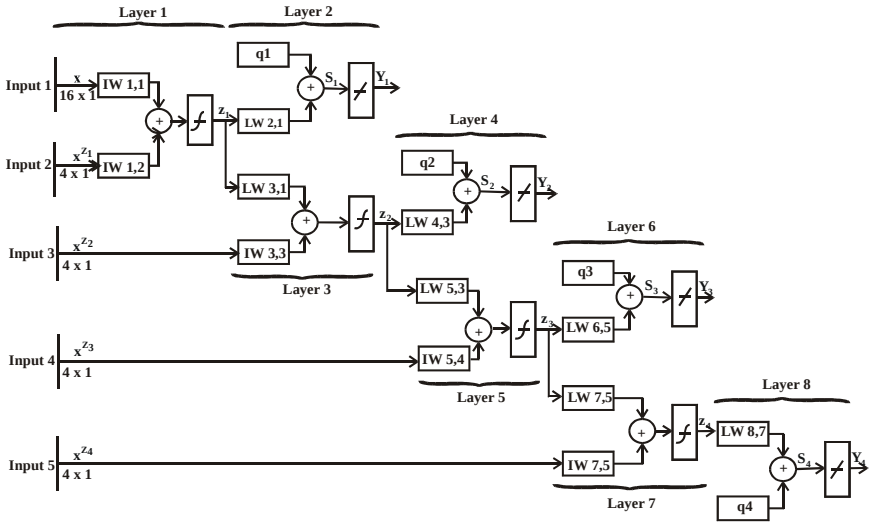

Рис. 4. Структурна схема імітаційної моделі ШНМ в пакеті Matlab

Оператор gensim(net) дає змогу одержати модель в пакеті Simulink (рис.5, 6).

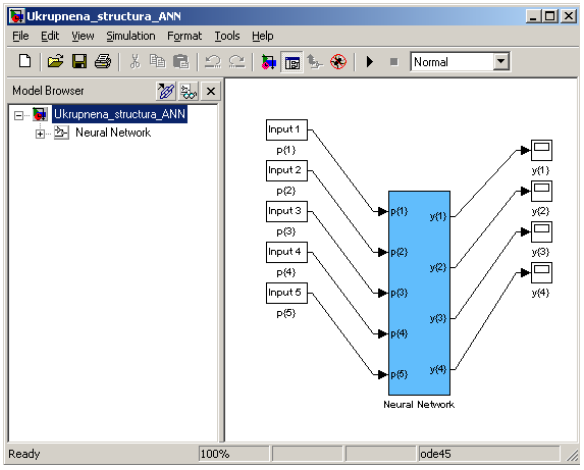

Рис. 5. Укрупнена структурна схема ШНМ в пакеті Simulink

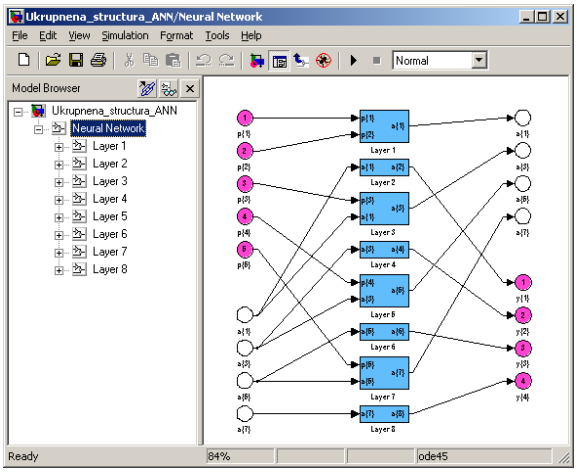

Рис. 6. Структурна схема шарів ШНМ в пакеті Simulink

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 [5, 6]. Для розрізнення значень вибірки, які відрізняються на порядок, було вирішено масштабувати навчальну вибірку за допомогою функції premnmx.

Максимальна похибка, яку було досягнуто при використанні комбінованого критерію якості навчання та масштабованої навчальної вибірки з 2250 векторів, становить 0.448359. Більшої точності навчання досягати 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).

Для тестування ШНМ було побудовано тестову вибірку з 200 векторів, яка також підлягала масштабуванню. Процес навчання і тестування [5, 6] різними алгоритмами навчання з використанням комбінованого критерію якості відображається на рис. 7, 8. На рисунках нижня крива відображає графік навчання, а верхня крива відображає графік тестування ШНМ.

Аналізуючи графіки навчання і тестування ШНМ, можна зробити висновок, що для навчання мережі найкраще підходить алгоритм навчання CGB на основі метода спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле та його модифікації (алгоритм навчання Флетчера-Рівса або алгоритм навчання Полака-Рібейри) з використанням комбінованого критерію оцінки якості навчання.

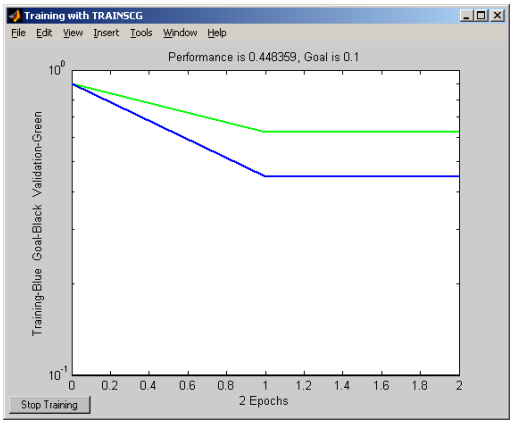

Рис. 7. Графіки навчання і тестування ШНМ за алгоритмом навчання SCG

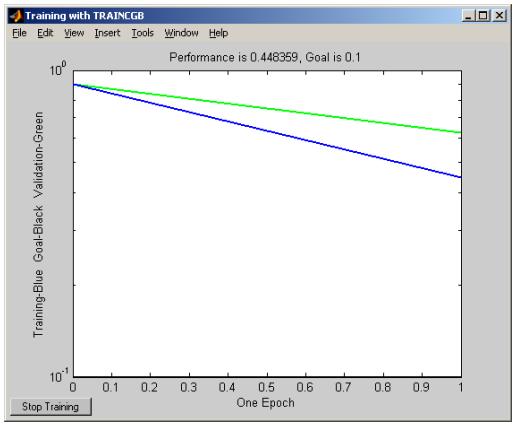

Рис. 8. Графіки навчання і тестування ШНМ за алгоритмом навчання CGB

Програмні засоби ідентифікації прихованих помилок програмного забезпечення

На рис.9 запропоновано структуру системи ідентифікації прихованих помилок програмного забезпечення [7], яка дозволяє користувачу на основі звіту про результати основного тестування одержати висновок про необхідність повторного тестування та про метод, яким рекомендується здійснювати повторне тестування.

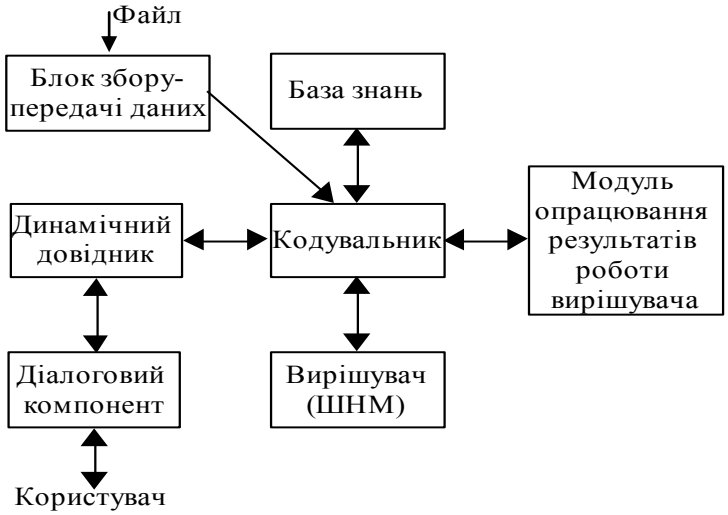

Рис. 9. Структурна схема системи ідентифікації прихованих помилок програмного забезпечення

Система ідентифікації прихованих помилок 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 таблиць 1-3 бази знань (додаток А), заповнення таблиці 5 бази знань (додаток А) вхідними даними 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 довіднику 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 таблиці 4 бази знань (додаток А) 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 та метод 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 та метод 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 саме: про наявність у програмному забезпеченні прихованих помилок та про метод, яким рекомендується здійснювати повторне тестування. 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 [7].

Після активізації системи користувачем на екрані з’являється наступне вікно (рис.10):

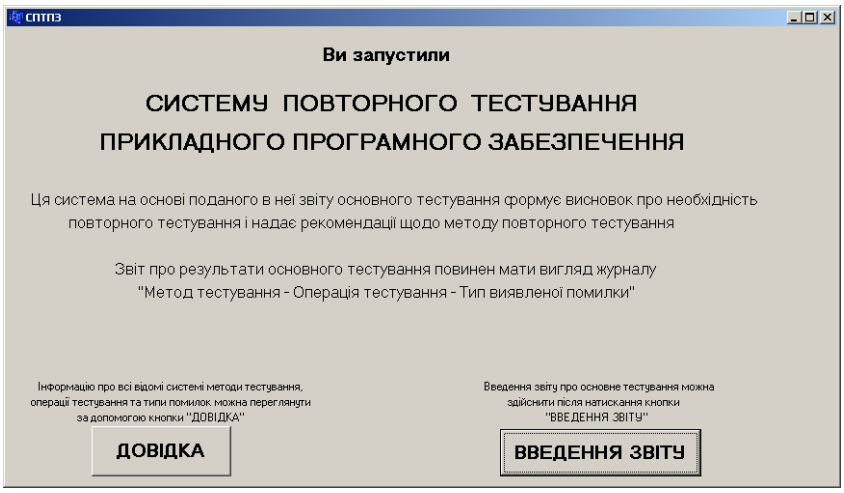

Рис.10. Перше діалогове вікно системи ідентифікації прихованих помилок ПЗ

При натисканні в першому діалоговому вікні кнопки “ВВЕДЕННЯ ЗВІТУ” користувач має змогу подати по рядку звіт в систему повторного тестування прикладного програмного забезпечення (рис.11).

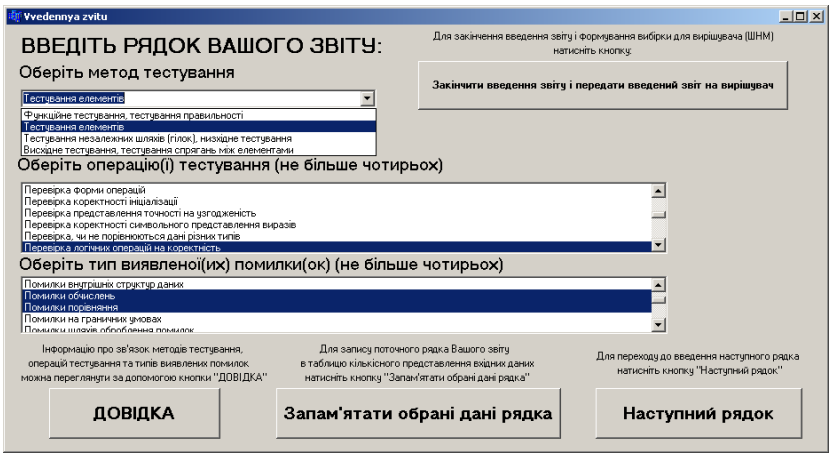

Рис.11. Введення рядка звіту

Висновки

Дістала подальшого розвитку концептуальна модель ідентифікації прихованих помилок програмного забезпечення, суть якої полягає у тому, що їх виявлення здійснюється після розроблення і налагодження ПЗ, де тестування програмного продукту здійснюється як часткова технологічна операція, тобто на впровадженому у життєвий цикл етапі повторного тестування, що може відповідати вхідному контролю.

Вперше одержано категорійну модель процесу повторного тестування ПЗ на базі ШНМ, котра відрізняється від відомих тим, що в ній враховується вплив прихованих помилок різних типів попередньої категорії на виникнення помилок наступної категорії. Це дає можливість оцінити сумарний вплив помилок цієї категорії на якість ПЗ і зробити висновок щодо необхідності повторного тестування ПЗ.

Удосконалено метод оцінки достовірності ідентифікації прихованих помилок ПЗ. Досліджено, що підвищення достовірності буде тим більше, чим більше буде виявлено прихованих помилок, що впливають на виникнення помилок на наступному рівні категорійності. У розглянутому в статті прикладі підвищення достовірності склало від 15 до 28%.

Література

1. *Калбертсон Р., Браун К., Кобб Г. Быстрое тестирование: Пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 384 с.*

2. *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.*

3. *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*

4. *Локазюк В.М., Поморова О.В., Говорущенко Т.О. Імітаційна модель системи повторного тестування програмного забезпечення // Вісник ХНУ – Хмельницький: ХНУ, 2006. – № 6, с.65-72*

5. *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*

6. *Govoruschenko T.O. Model of decision maker of repeated application software testing system // Радіоелектронні і комп'ютерні системи – Харків: НАУ "ХАІ", 2007 – № 7, с.191-198*

7. *Говорущенко Т.О. Реалізація та функціювання системи повторного тестування прикладного програмного забезпечення // Вісник Хмельницького національного університету – Хмельницький: ХНУ, 2007 - №2, т.2, с. 113-120*