**УДК 004.891.3: 004.3**

***Говорущенко Т.О.***

Хмельницький національний університет, м.Хмельницький, tat_yana@ukr.net,

kism@beta.tup.km.ua, Україна

# Інтелектуальна система визначення необхідності повторного тестування програмного забезпечення

В статье представлена интеллектуальная система определения необходимости повторного тестирования программного обеспечения, которая прогнозирует наличие скрытых ошибок в программном обеспечении после прохождения основного тестирования, а также определяет необходимость повторного тестирования. Приведены результаты работы предложенной системы.

## Вступ

Тестування програмного забезпечення є одним із основних методів забезпечення надійності програмного забезпечення (ПЗ). Найчастіше під *тестуванням* розуміють процес виявлення відмов за причини наявності помилок у програмах [1]. Найбільш поширеним трактуванням тестування є таке, коли *суть тестування* полягає у перевірці роботи програми за результатами їх виконання з використанням спеціально підібраних вихідних даних, які називають тестами [2]. Це, по суті, метод виявлення наявності помилок у програмах шляхом опрацювання тестових даних і порівняння одержаних при цьому результатів із розрахунковими.

Тестування здійснюється на різних етапах життєвого циклу ПЗ, зокрема, на етапах планування, проектування та кодування. На етапі планування проводиться аналіз і оцінка вимог до ПЗ як продукту, а також його характеристик і функційних можливостей. На етапі проектування коду програми ще немає, тестуванню піддаються вже повністю формалізовані і детально описані ідеї; особлива увага приділяється тому, чи відповідає проект вимогам документації, чи описує проект всі взаємозв'язки і передачу даних між модулями. На етапі кодування тестується вже готова програма.

## Постановка задачі

На сьогодні актуальними є задачі, що виникають при розробленні тестових програм: 1) прикладне програмне забезпечення стає об'ємним, тому що вирішує корпоративні задачі, складні аналітичні задачі, опрацьовує великі масиви даних, працює в багатьох напрямках і т.п.; 2) розроблене тестове програмне забезпечення вірно функціонує для прикладного ПЗ попередніх версій, але не завжди ефективне для сучасного прикладного ПЗ, тому що не враховує його особливостей.

До того ж, відмови ПЗ можуть бути зумовлені прихованістю помилок. В цьому випадку помилки проявляються тільки в окремих комбінаціях, що рідко

зустрічаються. Тому такі помилки виявляються тільки у процесі тривалої експлуатації ПЗ. Приховані помилки є найбільш небезпечними. Отже, *головною метою дослідження* є розроблення методів і засобів підвищення ефективності тестування програмного забезпечення за рахунок виявлення прихованих помилок ПЗ у процесі повторного тестування. Повторне тестування проводиться як окремий технологічний процес після розроблення і налагодження ПЗ.

# Концепція категорійності прихованих помилок

Щодо розподілу помилок ПЗ за їх видами і впливом на функціювання комп'ютерних систем, то в літературі [3] відомий їх розподіл за пріоритетами і категоріями. Розподіл помилок за пріоритетами здійснюється у відповідності до нормативів з визначення рівня серйозності помилок.

Нами проведено уточнення цього підходу щодо опису прихованих помилок. Всі приховані помилки розподілимо за видами на незначні (Н), помірні (П), серйозні (С) та катастрофічні (К) [4].

*Незначними (Н) прихованими помилками* вважатимемо такі, що не впливають на дії користувача, програмний продукт з їх наявністю придатний для використання.

*Помірними (П) прихованими помилками* вважатимемо такі, що впливають на дії користувача, але програмний продукт з їх наявністю придатний для використання з частковою втратою функційності.

*Серйозними (С) прихованими помилками* вважатимемо такі, що призводять до помилкових результатів, внаслідок чого програмний продукт непридатний до використання.

*Катастрофічними (К) прихованими помилками* вважатимемо такі, що призводять до спотворення інформації (даних), внаслідок чого програмний продукт непридатний до використання і намагання його використати призводить до відмови системи.

Незначним прихованим помилкам присвоїмо найнижчий рівень категорійності – перший. Помірним прихованим помилкам, відповідно, рівень 2; серйозним – рівень 3. Найвищим рівнем вважатимемо катастрофічний – рівень 4.

Певна кількість незначних помилок (Н′) призводить до появи окремих типів помірних помилок (П′), котрі, в свою чергу, призводять до появи певної кількості серйозних помилок (С′), а вони, відповідно, до - катастрофічних помилок (К′).

# Інтелектуальна система підтримки прийняття рішення про необхідність повторного тестування програмного забезпечення

Для вирішення проблеми підвищення ефективності тестування програмного забезпечення була розроблена інтелектуальна система визначення необхідності повторного тестування програмного забезпечення [5], на вхід якої подається звіт про основне тестування у вигляді журналу "Метод тестування – Операція тестування – Тип виявленої помилки". Ця система наведена на рис.1

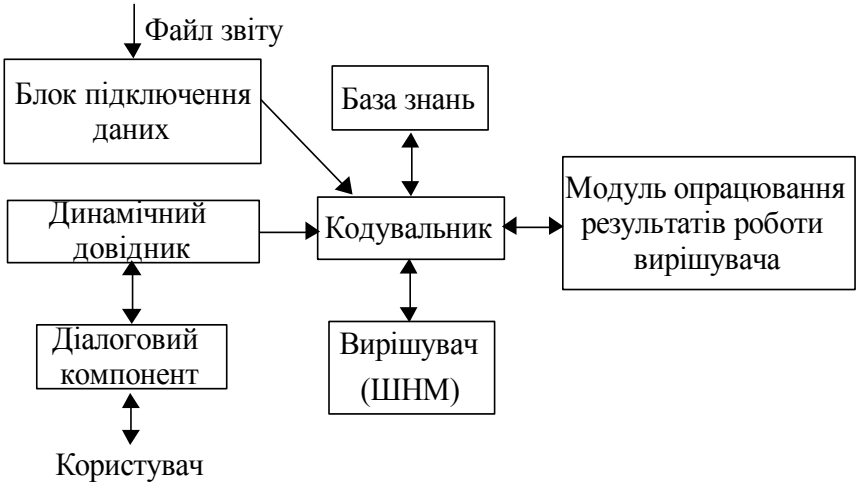


**Рис.1 Структурна схема інтелектуальної системи визначення необхідності повторного тестування програмного забезпечення**

На блок підключення даних подається файл користувача з результатами основного тестування, представленими у вигляді журналу "Метод тестування – Операція тестування – Тип виявленої помилки". Дані цього файлу передаються на кодувальник. Кодувальник здійснює перетворення вхідних даних з лінгвістичної форми у кількісну форму за допомогою таблиць 1, 2, 3 бази знань, в яких кожному методу, операції і типу помилки присвоєно номери.

**Таблиця 1 бази знань**

| **Номер** | **Метод тестування ПЗ** |
|---|---|
| 1 | Функційне тестування |
| 2 | Тестування елементів |
| 3 | Тестування незалежних шляхів (гілок) |
| … | … |

**Таблиця 2 бази знань**

| **Номер** | **Операція тестування ПЗ** |
|---|---|
| 1 | Перевірка, чи виконує програмна система (ПС) очікувані функції |
| 2 | Перевірка, чи виконує ПС поставлені вимоги |
| 3 | Перевірка, чи виконують модулі ПС очікувані функції |
| … | … |

**Таблиця 3 бази знань**

| **Номер** | **Тип виявленої помилки** |
|---|---|
| 1 | Помилки логічних умов |
| 2 | Помилки незалежних маршрутів програми |
| 3 | Помилки у гілках true, false для всіх логічних рішень |
| … | … |

Після цього кодувальник заповняє таблицю 4 бази знань вхідними даними у кількісній формі.

**Таблиця 4 бази знань**

| Метод тестування | Операція тестування | Тип виявленої помилки |
|---|---|---|
| № методу тестування *1*-го рядка вхідного звіту | № операції тестування *1*-го рядка вхідного звіту | № типу виявленої помилки *1*-го рядка вхідного звіту |
| … | … | … |
| № методу тестування *i*-го рядка вхідного звіту | № операції тестування *i*-го рядка вхідного звіту | № типу виявленої помилки *i*-го рядка вхідного звіту |
| … | … | … |
| № методу тестування *n*-го рядка вхідного звіту | № операції тестування *n*-го рядка вхідного звіту | № типу виявленої помилки *n*-го рядка вхідного звіту |

Далі кодувальник формує вхідні вектори вирішувача, в якості якого виступає ШНМ, наведена на рис.2.

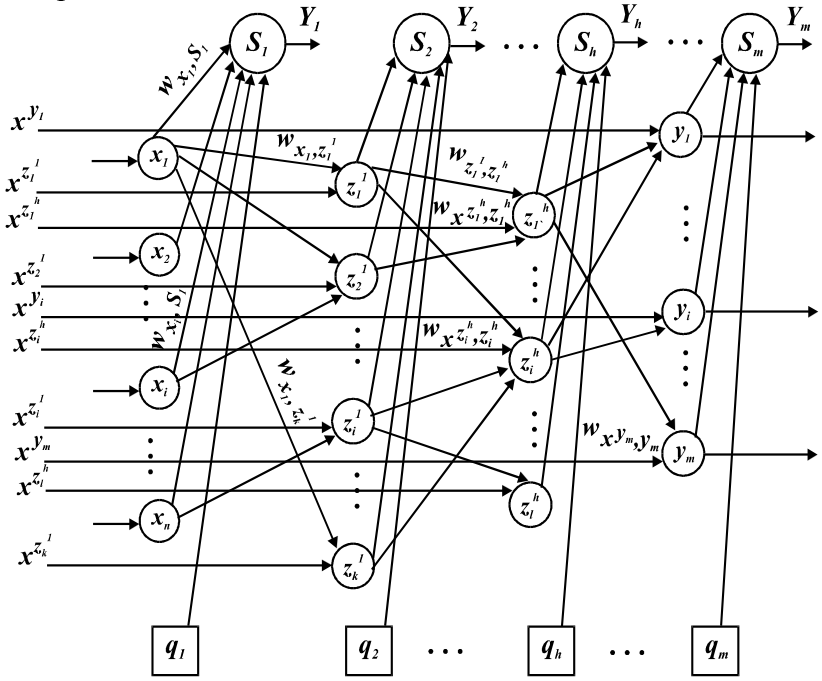

**Рис.2. Структура узагальненої складної штучної нейронної мережі, що відображає зв'язок помилок ПЗ різних категорій**

Для формування вхідного вектора ШНМ на основі $i$-го рядка таблиці 4 «одиниці» подаються на входи $q_{mn_i}$ ($mn_i$ - елемент першого стовпця $i$-го рядка таблиці 4), $x'_{on_i}$ ($on_i$ - елемент другого стовпця $i$-го рядка таблиці 4), $x_{pn_i}$ ($pn_i$ - елемент третього стовпця $i$-го рядка таблиці 4), тобто номери входів $q$, на які подаються „одиниці" відповідають номерам методів тестування, які були застосовані при основному тестуванні, номери входів $x'$ - номерам операцій тестування, номери входів $x$ - номерам типів виявлених під час основного тестування помилок. На всі інші входи ШНМ подаються «нулі».

Розв'язок поставленої задачі виявлення прихованих помилок грунтується на категорійній моделі процесу повторного тестування [4], в якій враховано важливість

кожного типу помилок, взаємний вплив типів помилок, нечіткість вихідних даних про наявні помилки і т.і., тому в якості вирішувача використовується штучна нейронна мережа (ШНМ).

ШНМ опрацьовує набір вхідних векторів 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 дані вирішувача (ШНМ) подаються на кодувальник, який здійснює перетворення результуючих векторів вирішувача з кількісної в лінгвістичну форму за допомогою таблиці 5 бази знань:

**Таблиця 5 бази знань**

| **Рівень категорійності прихованих помилок** | **Тип прихованих помилок** |
|---|---|
| 1 | Незначні (Н) |
| 2 | Помірні(П) |
| 3 | Серйозні (С) |
| 4 | Катастрофічні (К) |

Далі кодувальником здійснюється заповнення таблиці 6 бази знань результуючими даними у текстовій формі.

**Таблиця 6 бази знань**

| **Результуючі рівні категорійності** |
|---|
| Рівень категорійності, одержаний для *1*-го рядка вхідного звіту |
| … |
| Рівень категорійності, одержаний для $i$-го рядка вхідного звіту |
| … |
| Рівень категорійності, одержаний для $n$-го рядка вхідного звіту |

Після цього кодувальник здійснює передачу результатів роботи ШНМ у модуль опрацювання результатів роботи вирішувача.

Для опису правила формування висновку про необхідність повторного тестування програмного забезпечення введемо поріг $a_i \in A$, $A = \{ a_h \mid h = 1..4 \}$, де $a_h$ - поріг допустимої кількості помилок і важливості помилок різних типів одного виду, при перевищенні якого необхідно здійснювати повторне тестування з метою виявлення прихованих помилок цього виду, $h$ - кількість типів порогів, що змінюється від 1 до $s$, $s$ - кількість рівнів категорійності прихованих помилок). Тоді правило формування логічного висновку про необхідність повторного тестування матиме наступний вигляд.

**Правило 1.** Якщо відношення сумарного значення помилок $i$-го рівня категорійності до загальної кількості виявлених під час основного тестування помилок перевищує поріг $a_i$, то повторне тестування здійснювати необхідно.

Модуль опрацювання результатів роботи вирішувача на основі таблиці 6 та розділу правил генерує висновок про необхідність повторного тестування, який передається через діалоговий компонент користувачу.

# Результати роботи інтелектуальної системи визначення необхідності повторного тестування програмного забезпечення

Розглянемо приклад реалізації інтелектуальної системи визначення необхідності повторного тестування програмного забезпечення. Для навчання ШНМ і для формування вхідних даних системи в даному прикладі досліджувались прикладні програми з відкритим програмним кодом з набору програм Examples в пакеті Borland C++ Builder 5.0. В них штучно вносились помилки різних відомих системі типів, потім програма тестувалась відомими системі методами і операціями тестування ПЗ, в результаті чого формувались спочатку навчальна вибірка для ШНМ з 740 векторів (таблиця 1), а потім звіт про результати основного тестування з 80 рядків (таблиця 2).

**Таблиця 1. Навчальна вибірка для ШНМ**

| **Метод тестування ПЗ** | **Операція тестування ПЗ** | **Тип виявленої помилки** | **Рівень категорійності** |
|---|---|---|---|
| Низхідне тестування | Перевірка коректності роботи “заглушок” | Некоректна робота “заглушок” | Помірний |
| Висхідне тестування | Перевірка коректності об’єднання модулів в загальну структуру | Помилки об’єднання модулів в загальну структуру | Серйозний |
| Тестування правильності | Перевірка відповідності відомих функцій програми, одержаним | Відомі функції програми, не відповідають одержаним | Серйозний |
| … | … | … | … |

**Таблиця 2. Звіт про результати основного тестування ПЗ**

| **Метод тестування ПЗ** | **Операція тестування ПЗ** | **Тип виявленої помилки** |
|---|---|---|
| (1) | (2) | (3) |
| Тестування незалежних шляхів (гілок) | Перевірка правильності гілок True і False для всіх логічних рішень | Помилки логічних умов |
| Тестування елементів | Перевірка цілісності даних, що зберігаються | Помилки внутрішніх структур даних |

| (1) | (2) | (3) |
|---|---|---|
| Функційне тестування | Перевірка, чи виконує програма очікувані функції | Програма та її функціювання не відповідає наперед відомим функціям програми |
| … | … | … |

Після подання файлу із звітом (таблиця 2) в інтелектуальну систему визначення необхідності повторного тестування програмного забезпечення вирішувач (ШНМ) надавав в якості результату прогноз, що в програмі є 2 приховані помилки першого рівня категорійності (незначні), 4 приховані помилки другого рівня категорійності (помірні), 7 прихованих помилок третього рівня категорійності (серйозні), 5 прихованих помилок четвертого рівня категорійності (катастрофічні). Оскільки такі відношення сумарного значення помилок кожного рівня категорійності до загальної кількості виявлених під час основного тестування помилок для третього та четвертого рівнів категорійності перевищували поріг допустимої кількості прихованих помилок і важливості помилок свого рівня категорійності відповідно, то висновок був таким, що повторне тестування необхідне.

# Висновок

Запропонована інтелектуальна система визначення необхідності повторного тестування ПЗ дозволяє користувачу, надаючи в систему звіт про результати основного тестування, вирішити задачу прийняття рішення про необхідність повторного тестування, зокрема про наявність прихованих помилок та їх рівень категорійності.

In article the intelligent system of determination of repeated application software testing necessity, which prognoses hidden mistakes presence in software after basic testing and also determines necessity of repeated testing, is presented. Results of proposed system functioning are represented.

# Література

1. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие: Пер. с англ./Джон Макгрегор, Девид Сайкс. – К.: ООО “ТИД “ДС”, 2002. – 432 с.
2. Локазюк В.М., Савченко Ю.Г. Надійність, контроль, діагностика та модернізація ПК: Посібник. – К.: Видавничий центр “Академія”, 2004. – 376с.
3. Калбертсон Р., Браун К., Кобб Г. Быстрое тестирование: Пер. с англ. – М.: Издательский дом “Вильямс”, 2002. – 384 с.
4. 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
5. Говорущенко Т.О. Система повторного тестування програмного забезпечення // Радіоелектронні і комп’ютерні системи – Харків: НАУ “ХАІ”, 2005. - №4, с.120 – 126