

## МЕТОДИ ОРГАНІЗАЦІЇ ЗНАХОДЖЕННЯ ТА ВИПРАВЛЕННЯ ПОМИЛОК В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕНІ

*Стаття присвячена розробці програмно-алгоритмічних засобів для проведення оцінки надійності програмного забезпечення (ПЗ) на основі моделей надійності, що дозволяє проводити розрахунок характеристик надійності ПЗ.*

*В роботі проведено аналіз проблем знаходження надійності програмного забезпечення за допомогою моделей надійності. Встановлено, що досить упевнено прогнозувати рівень надійності функціонування ПЗ дуже важко. Проблема полягає в тому, що існуючі методи і моделі прогнозування надійності ПЗ не повною мірою придатні для практичного застосування. Аналізуючи проблеми програмного забезпечення зрозуміло, що визначити початкову кількість помилок в ПЗ досить важко (майже не можливо), тому в статті запропоновано метод зворотного розрахунку, перевагою якого є те, що в ньому не використовується припущення про початкову кількість помилок в ПЗ. Замість неї використовуються досить просто вимірювані характеристики ПЗ, такі як інтенсивність появи помилок і інтенсивність усунення помилок.*

*Для знаходження початкової кількості помилок в програмі запропоновано алгоритм пошуку помилок на області виділених даних.*

*Враховуючи те, що розроблена модель дозволяє вирішувати обернену задачу, в статті запропоновано метод пошуку початкової кількості помилок в програмі по початковій і кінцевій інтенсивності відмов.*

*В роботі проведено дослідження для періоду дослідної експлуатації ПЗ і можливість передачі системи в промислову експлуатацію. Для дослідження періоду дослідної експлуатації ПЗ в роботі запропоновано метод визначення часу, необхідного для зменшення кількості помилок в 2 рази. Проведено моделювання дослідних 180 днів промислової експлуатації. З метою оцінки надійності для систем типу «клієнт-сервер» на етапі тестування визначено такі важливі характеристики функціонування програмного комплексу, як: розрахунок поточного часу напрацювання повністю, розрахунок середнього часу напрацювання повністю за весь час моделювання роботи системи, розрахунок імовірності відмови ПЗ за одиницю часу, розрахунок коефіцієнта готовності. На основі розроблених методів та алгоритмів створено програму моделювання, що дозволяє, задаючи різні початкові умови, спостерігати поведінку надійності ПЗ в часі.*

*Ключові слова: надійність програмного забезпечення, моделі надійності, клієнт-серверна архітектура, програма-клієнт, область визначення вхідних даних.*

**Вступ.** Сьогодні в області машинної обробки інформації існують дві взаємозв'язані проблеми: вартість обробки інформації і ненадійність програмного забезпечення, які організують і забезпечують процес обробки інформації. При цьому перша проблема знаходиться в залежності від другої, оскільки висока вартість проектування, тестування і супроводу програм обробки інформації визначається передусім ненадійністю ПЗ.

Необхідність підвищення надійності програмного забезпечення обумовлена ще і тим, що нині ПЗ несе значно більше функціональне навантаження в вирішенні завдань керування, чим технічні засоби.

Основним засобом реалізації функцій обробки інформації і керування в сучасних корпоративних інформаційних системах (КІС) є програмне забезпечення (ПЗ). Істотною особливістю КІС являється безперервність процесів введення і обробки інформації, циклічний характер обчислювальних процесів. У зв'язку з цим найважливішою проблемою, що виникає при створенні КІС, є забезпечення високого рівня надійності їх функціонування.

Численні наукові публікації і накопичений досвід розробки програмних систем в Україні і за кордоном свідчать про те, що досить упевнено прогнозувати рівень надійності

функціонування ПЗ дуже важко. Проблема полягає в тому, що існуючі методи і моделі прогнозування надійності ПЗ не повною мірою придатні для практичного застосування.

У зв'язку з цим тематика досліджень, пов'язаних з підвищенням надійності програмного забезпечення, є актуальною.

На основі аналізу публікацій можна констатувати: на сьогодні відсутнє загальне вирішення проблеми надійності ПЗ і є багато приватних рішень, що не враховують такі істотні чинники як інтенсивність внесення і усунення помилок в програмі, час розробки ПЗ. Жодна з моделей не може вважатися достатньою для оцінки надійності.

Таким чином, сьогоднішній рівень розуміння проблеми надійності в основному якісний, що дозволяє розглядати програму як чорний ящик з даними, що поступають на вхід, зовнішні діями, а на виході - потік помилок, що усувається з великим або меншим успіхом.

**Постановка задачі.** Теорія надійності як наука отримала розвиток стосовно складних технічних систем. Необхідність і корисність контролю технічних компонент систем і систем в цілому, з метою перевірки відповідності їх поточних характеристик заданим, доведені практикою. У цьому плані виконана значна кількість робіт по надійності стосовно технічних систем, розроблено безліч моделей забезпечення розумними методами надійності складних систем і їх технічної готовності. Ці моделі у ряді випадків дозволяють не лише оцінювати показники надійності і готовності технічних систем і їх компонентів, але і дають можливість передбачати значення цих показників на основі накопиченого досвіду. Крім того, ряд моделей дозволяє на основі накопичених даних висловлювати припущення відносно режимів роботи, при яких найчастіше проявляються відхилення від нормального функціонування, а також про вживаний підхід до відновлення (ремонт) системи або її компонентів після збою.

**Мета статті.** Розробка програмно-алгоритмічних засобів для проведення оцінки надійності програмного забезпечення на основі відомих моделей надійності ПЗ, що дозволить проводити розрахунок характеристик надійності ПЗ (таких як, час напрацювання до відмови, коефіцієнт готовності, вірогідність відмови) і на основі цієї моделі прогнозувати зміну цих характеристик в часі та оптимізувати кількість помилок в програмному забезпеченні

**Виклад основного матеріалу дослідження.** Основною проблемою знаходження надійності ПЗ за допомогою моделей надійності є необхідність знати початкову кількість помилок ПЗ. Цю величину визначити досить важко (майже не можливо).

**Метод зворотного розрахунку.** Однією з переваг методу в порівнянні з іншими є те, що в ньому не використовується припущення про початкову кількість помилок  $N_0$  в ПЗ. Замість неї використовуються досить просто вимірювані характеристики ПЗ, такі як інтенсивність появи помилок і інтенсивність усунення помилок. Хоча запропонована модель надійності і не використовує цю величину, проте, можна скористатися її результатами для знаходження початкової кількості помилок в програмі  $N_0$  методом зворотного розрахунку. Це дозволить знайти такі характеристики надійності ПЗ, як час напрацювання до відмови, його вірогідність і час досягнення потрібної надійності за заданих початкових умов [9].

Програма написана в інтегрованому середовищі розробки програм Delphi із застосуванням об'єктно-орієнтованого (ОО) підходу, який забезпечує швидшу і компактнішу реалізацію алгоритму [3].

При дослідженні виконуються наступні кроки:

Пропонується розміщення  $E_r$  помилок на ОВД, розподілених на ній рівномірно.

Для кожного з  $K$  клієнтів пропонується на початку тільки один раз  $mk_i$  і  $\sigma k_i$ .

Далі ітеративно ( $M$  разів підряд) з кроком  $\Delta t$  для кожного клієнта.

Якщо клієнт справний, то він може звертатися із запитом до сервера з інтенсивністю  $\lambda_{зв}$ . Вірогідність звернення клієнта до сервера дорівнює  $1 - e^{-\lambda_{зв} \Delta t}$ .

У разі звернення клієнта до сервера формується випадкова величина  $x_i$ , розподілена за нормальним законом з параметрами  $mk_i$  і  $k_i$  - вхідні дані для запиту до сервера. Область, що використовується вхідними даними запиту від одного клієнта до сервера на ОВД, є величина  $x_i \pm \alpha/2$ .

Якщо в інтервал  $(x_i \pm \alpha/2)$  потрапляє хоч би одна помилка на ОВД, то вважається, що в клієтові виявлена помилка, і він виводиться з експлуатації для її виправлення одним з вільних програмістів. Якщо вільних програмістів немає, то несправний клієнт стає в чергу і чекає, коли один з програмістів звільниться.

Якщо в запиті клієнта до сервера помилки немає, то цей запит спрямовується серверу на обробку і відповіді. При цьому формується відповідь від сервера клієтові. Якщо в область  $(x_i \pm \alpha/2)$  потрапляє хоч би одна помилка зі списку помилок сервера, то вважається, що в сервері сталася помилка. В цьому випадку робота системи зупиняється і усі програмісти намагаються виправити цю помилку в сервері із швидкістю  $\lambda_{\text{випр}}$  кожен.

Вірогідність виправлення помилки одним програмістом рівна  $1 - e^{-\frac{\lambda_{\text{випр}} \Delta t}{s}}$ .

Отже якщо на певному кроці в клієтові виявлена помилка і є вільний програміст, то вільний програміст намагається виправити помилку в клієтові з вірогідністю  $1 - e^{-\frac{\lambda_{\text{випр}} \Delta t}{s}}$ . Якщо помилка виправляється, то вона видаляється зі списку помилок. Таким чином, ця помилка вже не може виникнути в інших клієнтах. При цьому якщо є клієнти, в яких була виявлена така ж помилка, то ці клієнти вважаються теж виправленими.

При виправленні помилки кожен програміст може внести нову помилку з вірогідністю рівна  $1 - e^{-\frac{\lambda_{\text{вн}} \Delta t}{s}}$ . Причому, якщо програміст вніс помилку в програму, то він може внести туди ще одну помилку з вірогідністю рівною квадрату вірогідності внесення попередньої помилки. Знову внесені помилки вносяться в список помилок. При цьому ці нові помилки не вважаються виявленими в клієтові або сервері, тобто якщо виявлена помилка виправляється, то клієнт або сервер вважається виправленим навіть, якщо при цьому були зроблені нові помилки.

За один часовий такт  $\Delta t$  формується сценарій обміну даними для усіх працюючих на цей момент часу клієнтів. Для несправних клієнтів або несправного сервера формується імовірнісний процес виправлення помилки в них.

В результаті формується  $M$  ітерацій і отримуємо одну реалізацію випадкових функцій  $m_1^c(t)$ ,  $m_2^c(t)$ ,  $m_2^{\bar{c}}(t)$  і  $\bar{p}(t)$  на часовому інтервалі  $M \cdot \Delta t$ .

Випробовування проводимо ще  $R$  разів і таким чином отримуємо  $R$  реалізацій випадкових функцій  $m_1^c(t), m_2^c(t), m_2^{\bar{c}}(t), \bar{p}(t)$ . Для кожного моменту часу  $t_j$  (для  $j = 1, \dots, M$ ) з кроком  $\Delta t$  знаходимо середнє статистичне для цих функцій і отримуємо середнє функцій  $\langle m_1^c(t) \rangle$ ,  $\langle m_2^c(t) \rangle$ ,  $\langle m_2^{\bar{c}}(t) \rangle$  і  $\langle \bar{p}(t) \rangle$ .

Алгоритм пошуку помилок на області виділених даних зображено на рисунку 1.

В процесі дослідження проводиться:

1. Розрахунок поточного часу напрацювання до відмови.
2. Розрахунок середнього часу напрацювання до відмови за весь час експерименту.
3. Розрахунок вірогідності відмови ПЗ в одиницю часу, як  $P = (\langle \text{об'єм запиту} \rangle \cdot \langle \text{кількість помилок в клієнтах і сервері} \rangle \cdot (\langle \text{кількість працюючих клієнтів} \rangle + 1) \cdot \langle \text{інтенсивність звернення} \rangle \cdot \langle \text{крок ітерації за часом} \rangle$ .

4. Розрахунок коефіцієнта готовності :  $K_g = 1 - \langle \text{час простою усієї програми} \rangle / \langle \text{час роботи} \rangle$ .



того, що кожну хвилину одна з програм-клієнтів отримує дані, отримуємо інтенсивність звернення до сервера - 1500 звернень в добу.

Експеримент показує, що такими початковими і кінцевими умовами відповідає ПЗ з 20-25 помилками на початку роботи і 10-14 помилок за 180 діб і коефіцієнтом готовності 0,9. Проведемо експеримент :

Початкові умови експерименту №1:

$K$  (к-ть програм-клієнтів) = 10;

$P$  (к-ть програмістів) = 3;

$\alpha$  (ширина запиту клієнта) = 0,000001;

$N_0$  (початкова кількість помилок) = 25;

$s$  (складність сервера) = 2;

$\Delta t$  (крок ітерації) = 0,0005 (доба);

$\lambda_{зв}$  (інтенсивність потоку звернень клієнта до сервера) = 1500/діб;

$\lambda_{випр}$  (інтенсивність потоку виправлення помилок) = 2/доби;

$p_{внес}$  (вірогідність внесення помилки при виправленні) = 0,1/діб;

$M$  (кількість ітерацій) = 360000;

Загальний час експерименту: 180 (доба);

$K$  (число експериментів) = 5.

Був отриманий результат (рис.2).

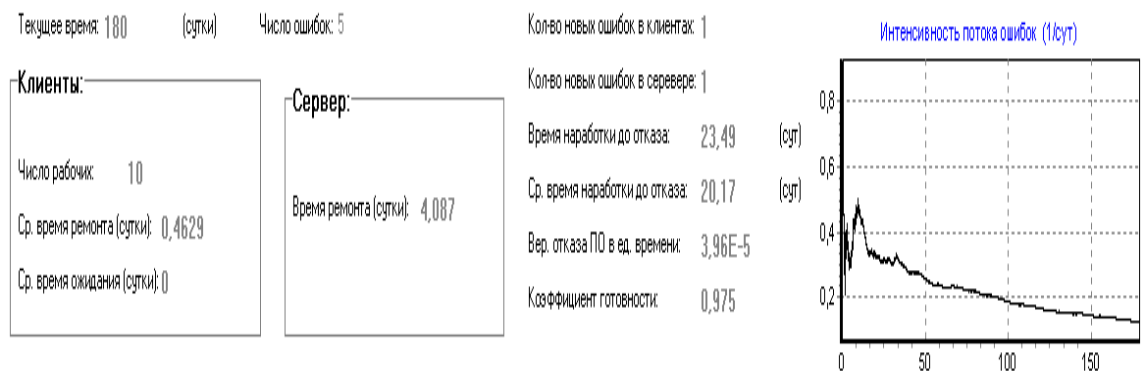


Рис.2. Результат пошуку кількості помилок для перших 180 днів експлуатації ПЗ

З отриманих даних видно, що після 180 днів дослідної експлуатації в системі залишаться приблизно 5 помилок і коефіцієнт готовності буде більше 0.95, а середній час напрацювання на відмову буде більше 20 діб. Це прийнятні значення для надійності такої системи і отже її можна передавати в промислову експлуатацію.

Моделювання наступних 180 днів промислової експлуатації показують, що за цей час буде виявлено і виправлено всього 3, 4 помилки, а коефіцієнт готовності досягне 0.99 і середній час напрацювання до відмови буде близько 60 днів (рис.3).

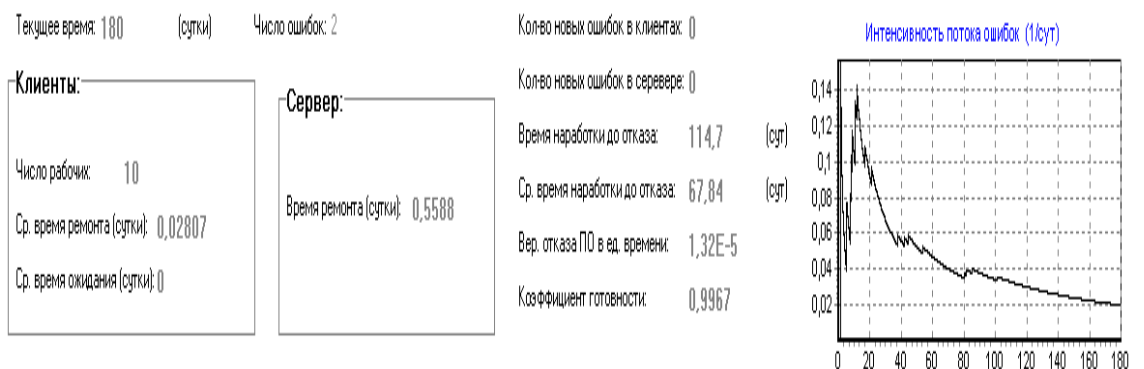


Рис. 3. Результат пошуку кількості помилок для наступних 180 днів експлуатації ПЗ

Для порівняння проведемо експеримент №2 для кількості помилок, що дорівнює сто і незмінних інших початкових умов (рис.4).

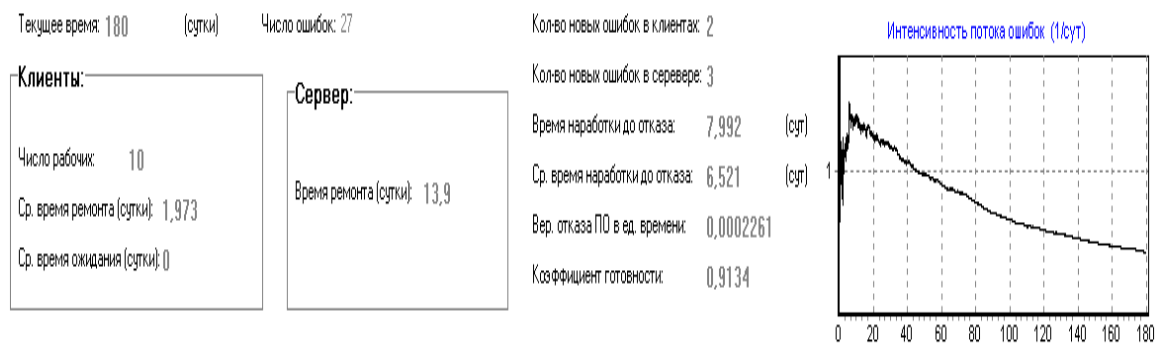


Рис. 4. Результат пошуку початкової кількості помилок для експерименту №2

З отриманих результатів видно, що до кінця дослідної експлуатації програми з такою початковою кількістю помилок через 180 днів ще залишиться близько 25 помилок, коефіцієнт готовності - 0.9, а середній час напрацювання до відмови - близько 6 діб, що говорить про низьку надійність ПЗ, внаслідок чого програма не готова до передачі в промислову експлуатацію і необхідно продовжити її дослідну експлуатацію ще на півроку.

Метод визначення часу, необхідного для зменшення кількості помилок до розрахункового рівня.

Знайдемо час, який необхідний для зменшення кількості помилок в 2 рази (рис.5-6). Початкові умови:

$$K \text{ (к-ть програм-клієнтів)} = 10;$$

$$P \text{ (к-ть програмістів)} = 3;$$

$$\alpha \text{ (ширина запиту клієнта)} = 0,0001;$$

$$N_0 \text{ (початкова кількість помилок)} = 100;$$

$$s \text{ (складність сервера)} = 2;$$

$$\Delta t \text{ (крок ітерації)} = 0,001 \text{ (доба)};$$

$$\lambda_{звр} \text{ (інтенсивність потоку звернень клієнта до сервера)} = 100/\text{діб};$$

$$\lambda_{випр} \text{ (інтенсивність потоку виправлення помилки)} = 0,2/\text{діб};$$

$$p_{внес} \text{ (вірогідність внесення помилки при виправленні)} = 0,005$$

$$M \text{ (кількість ітерацій)} = 200000;$$

Загальний час експерименту: 200 (доба);

$$K \text{ (число експериментів)} = 5.$$

За результатами дослідження отримуємо:  $T_{\frac{1}{2}} = 4$  дня, що є дуже оптимістичною оцінкою (рис.6).

**Параметры розыгрыша**

Кол-во программ-клиентов: 10  
Кол-во программистов: 3  
Доля от общей области данных (ООД) в одном запросе клиента: 0,0001

Блок оценки начального количества ошибок:  
Начальное кол-во ошибок: 10000  
Интенсивность начального потока ошибок в ПК: 100 (1/сутки)  
Оценить количество

Среднее число ошибок, которое будет найдено за время моделирования: 20000  
оптимальное время моделирования: 100

Коэффициент сложности сервера: 2  
Интенсивность потока обращений клиента к серверу: 100 (1/сутки)  
Интенсивность потока исправления ошибки: 0,2 (1/сутки)  
Вероятность внести ошибку при исправлении: 0  
Интенсивность внесения ошибки при исправлении: 0,2 (1/сутки)  
Шаг итерации: 0,001 (сутки)  
Кол-во итераций: 200000  
Кол-во розыгрышей: 5  
Общее время розыгрыша: 200 (сутки)

Принять параметры (F2)

Рис. 5. Форма для введения початкових параметрів дослідження

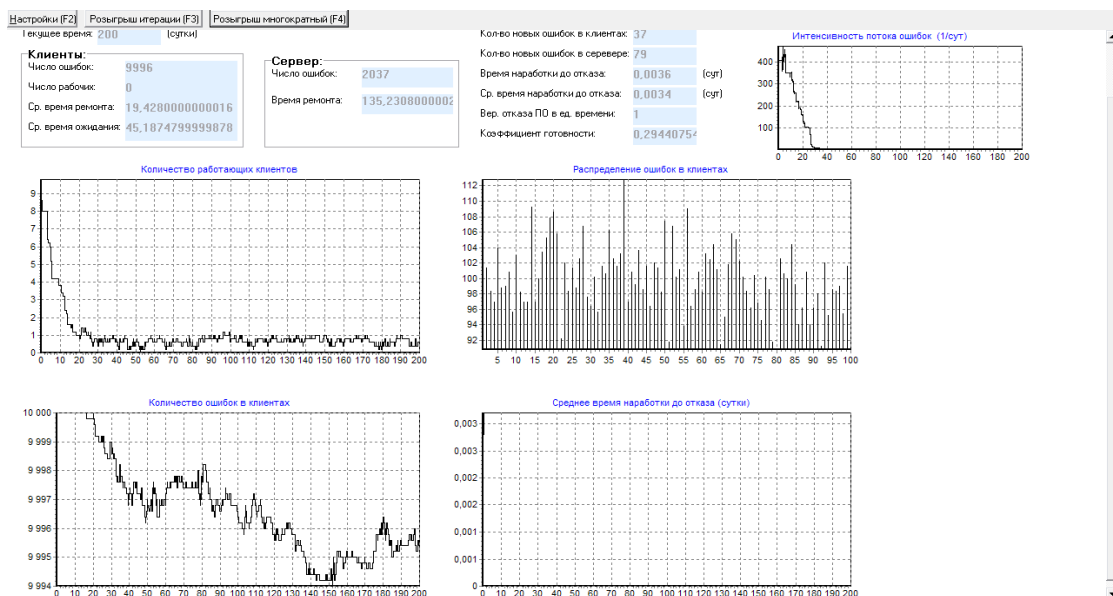


Рис. 6. Форма з результатами моделювання

**Висновки.** В роботі здійснено аналіз проблем знаходження надійності програмного забезпечення за допомогою моделей надійності. Встановлено, що упевнено прогнозувати рівень надійності функціонування ПЗ дуже важко. На сьогодні відсутнє загальне вирішення проблеми надійності ПЗ і є багато приватних рішень, що не враховують такі істотні чинники як інтенсивність внесення і усунення помилок в програмі, час розробки ПЗ. Жодна з моделей не може вважатися достатньою для оцінки надійності.

Таким чином, за результатами проведеного дослідження в роботі запропоновані алгоритмічно реалізовані методи організації знаходження та виправлення кількості помилок в програмному забезпеченні.

Дослідження проводились за допомогою програми, що написана в інтегрованому середовищі розробки програм Delphi із застосуванням об'єктно-орієнтованого (ОО) підходу, який забезпечує швидшу і компактнішу реалізацію алгоритму.

Таким чином, створена програма моделювання дозволяє, задаючи різні початкові умови, спостерігати поведінку надійності ПЗ в часі. Це дозволяє використати її для

вирішення оптимізаційних завдань (наприклад, пошуку оптимальних ресурсів для досягнення заданого рівня надійності, оптимальної інтенсивності тестування при заданих характеристиках клієнт-сервера і кількості програмістів). Зокрема, це дозволяє знайти початкову кількість помилок ПЗ.

Модель і програму моделювання можна рекомендувати використати при розробці і супроводі ПЗ, коли рівень надійності має бути високим, а досягти і підтвердити його не просто.

#### ЛІТЕРАТУРА:

1. Бабій І.М. Аналіз програмно– алгоритмічних засобів проведення оцінки надійності програмного забезпечення / І.М. Бабій, О.С. Ленков, О.В. Огневий // Зб.наукових праць Військового інституту Київського НУ ім.Тараса Шевченка – К.:ВІКНУ, 2016.–Вип.№52 – С86–93.
2. Барсегян А. Анализ данных и процессов / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2009. – 512 с.: ил.
3. Промыслов В.Г. Оценка надежности программного обеспечения на различных этапах жизненного цикла сложных программ/ В.Г. Промыслов, А.В. Антонов, С.И. Масолкин, А.С. Степанянц // Труды V Междунар. конф. "Идентификация систем и задачи управления" (SICPRO–2006). – С.1300–1304.
4. Лисс В.А. Математические модели надежности программного обеспечения распределенных систем / В.А. Лисс // Известия СПбГТУ "ЛЭТИ". Сер. "Информатика, управление и компьютерные технологии". – 2005. – Вып.2. – С.26–32.
5. Липаев В.В. Функциональная безопасность программных средств / В.В. Липаев. – М.: СИНТЕГ, 2004. – 340 с.
6. Липаев В.В. Надежность и функциональная безопасность комплексов программ реального времени. – М: 2013[Е лектронний ресурс]. – Режим доступу: [http://www.ispras.ru/preprints/docs/prep\\_26\\_2013.pdf](http://www.ispras.ru/preprints/docs/prep_26_2013.pdf)
7. Надежность программного обеспечения информационных систем [Електронний ресурс]. – Режим доступу: <http://baumanki.net/lectures/10-informatika-i-programmirovaniye/350-nadezhnost-informacionnyh-sistem/4741-13-nadezhnostprogrammogo-obespecheniya.html>.
8. Огневий О.В. Консолідація інформації як найважливіший чинник ефективного керування підприємством / О.В. Огневий, А.М. Огнева // Вісник ХНУ. – Хмельницький, №4, ТЗ(214), 2014. – С.109–113.
9. Огневий О.В. Методи організації інформації в інформаційно-аналітичних системах / О.В. Огневий, А.М. Огнева, Д.В. Зайцев, О.В. Банзак // Зб. наукових праць Військового інституту Київського НУ ім.Тараса Шевченка.К.:ВІКНУ,2015. – Вип.№49. – С.68–74.
10. Огневий О.В. Принципи побудови розподілених автоматизованих навчальних систем / О.В. Огневий, І.М. Бабій. // Тези доповідей Всеукраїнської науково-практичної конференції молодих вчених, ад'юнктів, слухачів, курсантів і студентів “Молодіжна військова наука у Київському національному університеті імені Тараса Шевченка” / за заг. редакцією В.В. Балабіна. – К. : ВІКНУ, 2016. – С.51.
11. Огневий О.В. Проблеми досліджень надійності програмного забезпечення / О.В. Огневий, І.М. Бабій. // Тези доповідей XII Міжнародної науково-практичної конференції «Військова освіта і наука: сьогодення та майбутнє» / за заг. редакцією В.В. Балабіна. — К.: ВІКНУ, 2016. — С.21–22.
12. Поморова О.В. Сучасні проблеми оцінювання якості програмного забезпечення / О.В. Поморова, Т. О. Говорущенко // Радіоелектронні та комп'ютерні програми : наук.-техн. журнал. – Національний лісотехнічний університет України 294 Серія економічна Харків : Вид-во НАУ ім. М.Є. Жуковського "Харківський авіаційний інститут". – 2013. – № 5. – С. 319-327.

#### REFERENCES:

1. Babiy I.M. Analiz prohramno– alhorytmichnykh zasobiv provedennya otsinky nadiynosti prohramnoho zabezpechennya / I.M. Babiy, O.S. Lyenkov, O.V. Ohnyevyy // Zb.naukovykh prats' Vys'kovoho instytutu Kyuyivs'koho NU im.Tarasa Shevchenka – K.:VIKNU, 2016.–Vyp.#52 – S86–93.

2. Barsehyan A. Analiz dannykh y protsessov / A. A. Barsehyan, M. S. Kupryyanov, Y. Y. Kholod, M. D. Tess, S. Y. Elyzarov. – 3-e yzd., pererab. y dop. – SPb.: BKhV-Peterburh, 2009. – 512 s.: yl.
3. Промыслов В.Н. Otsenka nadezhnomy prohrannnoho obespechenyya na razlychnykh etapakh zhyznennoho tsykla slozhnykh prohrann / V.H. Промыслов, A.V. Antonov, S.Y. Masolkyn, A.S. Stepanyants // Труды V Mezhdunar. konf. "Ydentyfikatsyya system y zadachy upravlenyya" (SICPRO–2006). – S.1300–1304.
4. Lyss V.A. Matematycheskiye modely nadezhnomy prohrannnoho obespechenyya raspredelennykh system / V.A. Lyss // Yzvestyya SPbHTU "LЭTY". Ser. "Ynformatyka, upravlenye y komp'yuternyye tekhnolohyy". – 2005. – Выр.2. – S.26–32.
5. Lypaev V.V. Funktsyonal'naya bezopasnost' prohrannnykh sredstv / V.V. Lypaev. – M.: SYNTEH, 2004. – 340 s.
6. Lypaev V.V. Nadezhnost' y funktsyonal'naya bezopasnost' kompleksov prohrann real'noho vremeny. – M: 2013[E lektronnyy resurs]. – Rezhym dostupu: [http://www.ispras.ru/preprints/docs/prep\\_26\\_2013.pdf](http://www.ispras.ru/preprints/docs/prep_26_2013.pdf).
7. Nadezhnost' prohrannnoho obespechenyya ynformatsyonnykh system [Elektronnyy resurs]. – Rezhym dostupu: <http://baumanki.net/lectures/10-informatika-i-programmirovaniye/350-nadezhnost-informacionnyh-sistem/4741-13-nadezhnost-programmnogo-obespecheniya.html>.
8. Ohnyevyy O.V. Konsolidatsiya informatsiyi yak nayvazhlyvishyy chynnyk efektyvnoho keruvannya pidpryyemstvom / O.V. Ohnyevyy, A.M. Ohneva // Visnyk KhNU. – Khmel'nyts'kyy, #4, T3(214), 2014. – S.109–113.
9. Ohnyevyy O.V. Metody orhanizatsiyi informatsiyi v informatsiyno-analitychnykh systemakh / O.V. Ohnyevyy, A.M. Ohneva, D.V. Zaytsev, O.V. Banzak // Zb. naukovykh prats' Viys'kovoho instytutu Kyiv's'koho NU im.Tarasa Shevchenka.K.:VIKNU,2015. – Выр.#49. – S.68–74.
10. Ohnyevyy O.V. Pryntsypy pobudovy rozpodilynykh avtomatyzovanykh navchal'nykh system / O.V. Ohnyevyy, I.M. Babiy. // Tezy dopovidey Vseukrayins'koyi naukovo-praktychnoyi konferentsiyi molodykh vchenykh, ad'yunktiv, slukhachiv, kursantiv i studentiv “Molodizhna viys'kova nauka u Kyiv's'komu natsional'nomu universyteti imeni Tarasa Shevchenka” / za zah. redaktsiyeyu V.V. Balabina. – K. : VIKNU, 2016. – S.51.
11. Ohnyevyy O.V. Problemy doslidzhen' nadiynosti prohrannnoho zabezpechennyya / O.V. Ohnyevyy, I.M. Babiy. // Tezy dopovidey KhII Mizhnarodnoyi naukovo-praktychnoyi konferentsiyi «Viys'kova osvita i nauka: s'ohodennyya ta maybutnye» / za zah. redaktsiyeyu V.V. Balabina. — K.: VIKNU, 2016. — S.21–22.
12. Pomorova O.V. Suchasni problemy otsinyuvannya yakosti prohrannnoho zabezpechennyya / O.V. Pomorova, T. O. Hovorushchenko // Radioelektronni ta komp'yuterni prohrany : nauk.-tekhn. zhurnal. – Natsional'nyy lisotekhnichnyy universytet Ukrayiny 294 Seriya ekonomichna Kharkiv : Vyd-vo NAU im. M.Ye. Zhukovskoho "Kharkiv's'kyy aviatsiyyny instytut". – 2013. – # 5. – S. 319-327.

*Ph.D. Ognjevji O.V., Ph.D. Ogneva A.M.*

#### **METHODS OF ORGANIZATION OF INTRODUCTION AND CORRECTION OF ERRORS IN SOFTWARE PROVIDED**

The article is devoted to the development of software-algorithmic tools for the evaluation of the reliability of software (software) on the basis of reliability models, which allows calculating the reliability characteristics of software. In the work, an analysis of the problem of finding software reliability with reliability models has been carried out. It is found that it is very difficult to predict the level of reliability of software functioning quite confidently. The problem lies in the fact that existing methods and models of prediction of software reliability are not fully suitable for practical application.

Analyzing software problems it is clear that determining the initial number of errors in the software. It is difficult therefore the article proposes a method of inverse calculation, the advantage of which is that it does not use the assumption of the initial number of errors in the software. Instead, it uses rather simple measurable features of the software, such as the intensity of errors and the intensity of error correction. To find the initial number of errors in the program proposed algorithm for finding errors in the area of selected data.

Given that the developed model allows solving the inverse problem, the article proposes a method for finding the initial number of errors in the program for the initial and final failure rates.

In order to study the period of experimental exploitation of software in work, the method of determining the time required to reduce the number of errors by 2 times. In the article the simulation of

the experimental 180 days of industrial exploitation was conducted. In order to assess the reliability of "client-server" systems at the testing stage, the following important characteristics of the software system's functioning are determined: calculation of the current time of complete development, calculation of the average time of elaboration for the entire time of modeling of the system, calculation of the probability of failure of the software per unit time, calculation of the readiness factor. On the basis of the developed methods and algorithms, a simulation program was created, which allows, by specifying different initial conditions, to observe the behavior of software reliability in time.

Keywords: reliability of software, reliability model, client-server architecture, client program, field of definition of input data.

**Огнєвий Олександр Вікторович**, кандидат технічних наук, доцент кафедри кібербезпеки та комп'ютерних систем і мереж Хмельницького національного університету (Хмельницький, Україна).

**Огнева Алла Миколаївна**, кандидат технічних наук, доцент кафедри обліку, аудиту та оподаткування Хмельницького національного університету (Хмельницький, Україна).

**Ognjevyy Oleksandr**, Candidate of Technical Sciences, assistant professor of the Department of Cybersecurity and Computer Systems and Networks of the Khmelnytsky National University (Khmelnytsky, Ukraine).

**Ogneva Alla**, Candidate of Technical Sciences, Associate Professor, Department of Accounting, Auditing and Taxation of Khmelnytsky National University (Khmelnytsky, Ukraine).