УДК 004.005 **DOI: 10.20998/2078-9130.2025.1.332551**

Д. В. БРЕСЛАВСЬКИЙ, М. А. БОРОДІН, М О. ГРОШЕВИЙ, О. А. ТАТАРІНОВА

ВИКОРИСТАННЯ РІВНЯННЯ АВТОМОДЕЛЬНОГО ТИПУ ДЛЯ МОДЕЛЮВАННЯ НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, РЕАЛІЗОВАНОГО ЗА ДОПОМОГОЮ ХМАРНИХ ТЕХНОЛОГІЙ

Надано опис процесу тестування та налагодження програмного забезпечення та комплексу для реалізації скінченноелементного аналізу у хмарному середовищі. Аналізуються типи та причини виникнення помилок при налаштуванні та виконанні програмного коду, серед яких: помилки, що виникають під час розгортання інфраструктури та перешкоджають створенню необхідних ресурсів хмарного середовища; помилки в конфігурації інфраструктури, які не заважають роботі середовища, але роблять його використання неефективним або ненадійним; помилки, пов'язані з доступом до ресурсів в хмарному середовищі; неефективні рішення, що не заважають роботі, але знижують її ефективність. Описано інструменти для забезпечення роботи у хмарному середовищі, серед яких Azure Resource Manager, Terraform Cloud, GitHub Actions та інші. Використано архітектуру типу «головний сервіс – працюючий сервіс». Описано вибрану за результатами досліджень конфігурацію системи. Розглянуто питання керування хмарними ресурсами. За аналізом експериментально отриманої залежності числа помилок та відмов від часу запропоновано математичну модель для прогнозування надійності роботи програмних засобів. Модель побудовано на базі інтегрування за часом диференційного рівняння автомодельного типу для параметру відмов, близькість якого до нуля у певний інтервал часу й забезпечує прогноз потрібної надійності програмного засобу. Обговорюється процедура ідентифікації параметрів моделі зі степеневою залежністю від числа відмов. Показано задовільну здатність рівняння автомодельного типу до прогнозування надійності програмних засобів розглянутої конфігурації при порівнянні з реальними даними тестування. Запропоновано використання розробленої математичної моделі прогнозування надійності для її застосування для інших типів програмних систем, що використовують хмарні технології.

Ключові слова: надійність, відмова, помилка, хмарні технології, програмне забезпечення, тестування, автомодельне диференційне рівняння.

D. BRESLAVSKY, M. BORODIN, M. HROSHEVYI, O. TATARINOVA

USING AN AUTOMODEL TYPE EQUATION FOR MODELLING THE RELIABILITY OF SOFTWARE IMPLEMENTED BY USE OF CLOUD TECHNOLOGIES

A description of the testing and debugging process of a software for implementing finite element analysis in a cloud environment is provided. The types and causes of errors during configuration and execution of the program code are analyzed, including: errors that occur during infrastructure deployment and prevent the creation of the necessary cloud environment resources; errors in the infrastructure configuration that do not interfere with the operation of the environment, but make its use ineffective or unreliable; errors related to access to resources in the cloud environment; ineffective solutions that do not interfere with operation, but reduce its efficiency. Tools for ensuring operation in a cloud environment are described, including Azure Resource Manager, Terraform Cloud, GitHub Actions and others. The "main - worker" architecture is used. The selected system configuration is based on the research results is described. The issue of cloud resource management is considered. Based on the analysis of the experimentally obtained dependence of the number of errors and failures on time, a mathematical model is proposed for predicting the reliability of software operations. The model is built on the basis of time integration of a differential equation of the automodel type for the failure parameter, the proximity of which to zero in a certain time interval provides a forecast of the required reliability of the software tool. The procedure for identifying model parameters with a power dependence on the number of failures is discussed. The satisfactory ability of the automodel type equation to predict the reliability of software tools of the considered configuration is shown when compared with real testing data. The use of the developed mathematical model of reliability prediction for its application to other types of software systems using cloud technologies is proposed.

Key words: reliability, failure, error, cloud technologies, software, testing, automodel differential equation.

Вступ. Проведення комп’ютерного моделювання складних технічних процесів з використанням сучасних чисельних методів потребує підвищеного рівня застосування обчислювальної техніки. Одним з засобів рішення виникаючих проблем є проведення обчислень з використанням хмарних технологій [1-5]. Використання хмарних технологій обумовлено потребою в масштабованості обчислювальних ресурсів для проведення скінченноелементного аналізу з високими вимогами до пам’яті та процесорної потужності, що унеможливлює використання звичайного персонального комп’ютера або локального сервера. При цьому, завдяки необхідності рішення великої кількості технічних питань, реалізація у хмарі нових розроблених програмних засобів може зустрітись з певною кількістю проблем, пов’язаних з помилками налаштування, відмовами тощо. Завдяки цьому забезпечення надійності реалізованих у хмарі програмних рішень потребує розробки нових методів для її моделювання та прогнозування [2]. Вони будуються як на основі відомих, запропонованих та використаних раніше методів оцінювання надійності програмних систем [6], так й з застосуванням нових підходів. В роботі [5] розглянуто технології, що використовуються для проведення хмарних обчислень, аналізуються фактори, що впливають на їхнє застосування, насамперед такі, як вартість та швидкість. Проведено порівняння інструментів, що застосовуються для використання систем хмарних обчислень.

Для прогнозування надійності програмних рішень є можливість використання відомих класичних методів та математичних моделей [7-9], але й виконується розробка спеціалізованих засобів.

Авторами роботи [10] підкреслюється, що для критично важливих систем необхідно гарантувати безпеку та надійність. В роботі запропоновано новий метод планування, який називається «зіставлення та багатораундове розподілення» (MMA). Його використано для оптимізації тривалості виконання та загальної вартості всіх поданих завдань з урахуванням обмежень безпеки та надійності. Перший етап методу полягає у визначенні найбільш відповідних ресурсів для виконання завдань із забезпеченням вимог до продуктивності, безпеки та надійності у багатохмарному середовищі. Другий етап пов'язано з ітераційним виконанням завдань для оптимізації часу та вартості їхнього виконання шляхом мінімізації дисперсії очікуваного часу виконання.

Роботу [11] присвячено забезпеченню надійності хмарних систем при балансуванні навантаження у реальних ситуаціях неоднорідного розподілу ресурсів. У даній роботі використано алгоритм оптимізації, заснований на здатності ресурсів підтримувати належне балансування навантаження. Основу алгоритму побудовано на пошуку непрацюючих або зайнятих вузлів з подальшим обчисленням функції придатності.

Основні розроблені та реалізовані на теперішній час підходи та методи до забезпечення та прогнозування надійності програмних систем, що працюють у хмарних середовищах, аналізуються в оглядах [12-14]. У більшості робіт увагу приділено технічним питанням реалізації систем. Методи їхнього математичного прогнозування представлено менш детально. У даній роботі запропоновано метод математичного моделювання надійності програмного засобу, обчислення за допомогою якого виконуються з використанням хмарних технологій, із застосуванням рівняння автомодельного типу.

Метою цієї роботи є розробка математичної моделі прогнозування надійності програмного забезпечення на основі програмного комплексу, реалізованого з використанням хмарних технологій, на основі аналізу процесу виявлення помилок під час реального ходу тестування та налагодження. Запропонована модель орієнтована на використання у практиці інженерного розгортання програмного забезпечення в умовах обмеженої статистичної інформації, з метою оцінки моменту досягнення заданого рівня надійності.

Опис процесу налаштування та тестування для роботи у хмарному середовищі. Проєктування інженерних програмних комплексів є досить складним завданням, що включає до себе розробку математичних моделей, реалізацію алгоритмів та програмних модулів, їхнє тестування [15]. Ці задачі є досить добре розробленими та такими, що мають необхідні методики оцінювання надійності. В даній роботі розглянемо оцінювання надійності програмного забезпечення на основі програмного комплексу (ПК), розгорнутого для застосування з використанням хмарних технологій [16]. Комплекс призначено для скінченноелементного аналізу при загальному тривимірному напружено-деформованому стані з використанням скінченного елементу у формі чотирьохвузлового тетраедру.

Під час налаштування хмарного середовища для реалізації обчислень з використанням ПК було виявлено та виправлено наступні типи помилок:

а) помилки, що виникають під час розгортання інфраструктури, які перешкоджають створенню необхідних ресурсів хмарного середовища;

б) помилки в конфігурації інфраструктури, які не заважають роботі середовища, але роблять його використання неефективним або ненадійним;

в) помилки в ПК, пов'язані з доступом до ресурсів в хмарному середовищі;

г) неефективні рішення в ПК, що не заважають роботі, але знижують її ефективність.

Розглянемо спочатку опис розгортання інфраструктури для обчислень. Її було розгорнуто за допомогою сервісу Microsoft Azure Kubernetes Service [17]. У процесі реалізації було застосовано підхід «infrastructure as a code» із використанням засобів Terraform та Helm. У процесі створення Terraform- і Helm-маніфестів були виявлені конфігураційні похибки, що призводили або до блокування процесу розгортання, або до некоректного функціонування інфраструктури.

Розглянемо помилки доступу до Azure Resource Manager [18]. Для автоматизації розгортання Terraform [19] інфраструктури було використано GitHub Actions [20] та Terraform Cloud. Для коректного функціонування процесу, GitHub Actions повинен мати доступ до ініціалізації розгортання у середовищі Terraform Cloud, тоді як Terraform Cloud, у свою чергу, потребує доступу до Azure Resource Manager для створення всіх необхідних ресурсів. Для інтеграції GitHub Actions із Terraform Cloud використовується API-токен, застосування якого не викликало ускладнень. В свою чергу, для взаємодії з Azure Resource Manager використовуються механізми автентифікації та авторизації Microsoft Entra ID, рольова модель доступу (RBAC) і протокол OpenID Connect. Під час налаштування доступу для OpenID Connect-клієнтів виникли труднощі з конфігурацією прав доступу, що призвели до обмеження у створенні ресурсів із боку Terraform Cloud.

У ході формування опису ресурсів за допомогою Terraform були виявлені невідповідності API та логічні порушення у визначенні залежностей між ресурсами. Це призводило до передчасного запуску створення нових ресурсів до завершення створення залежних від них ресурсів, необхідних для їхньої коректної роботи.

Проаналізуємо проблему з недоступністю Azure ресурсів. За замовченням, для Azure підписки доступна тільки невелика кількість ресурсів. Існують обмеження як по кількості ресурсів, так і по сумарній кількості виділених потужностей (кількості пам'яті, кількості ядер процесора, тощо). У зв'язку з тим, що інфраструктура використання ПК вимагає достатньо великої кількості ресурсів, ці квоти були перевищені,

що призвело до неможливості створити необхідні ресурси. Для вирішення цієї проблеми довелось зробити декілька запитів до підтримки Microsoft Azure та дочекатися їхнього розгляду.

Наступний тип отриманих помилок – це помилки доступу до зовнішніх ресурсів. Домен, який використовувався для створення суб-доменів для програм в кластері обслуговується компанією Cloudflare. Тож під час виконання Terraform розгортання в домені створюється один NS запит, що веде до Azure DNS Zone для подальшого створення суб-доменів. Для цього було необхідно налаштувати доступ Terraform Cloud до API Cloudflare. Під час конфігурації доступу виявлено обмеження прав, що спричинило збої у розгортанні інфраструктури через API Cloudflare. Для створення SSL сертифікатів було використано сервіс ACME (Let's Encrypt). Хоч ACME і не потребує авторизації, сервіс має обмеження кількості успішних та неуспішних запитів SSL сертифікатів. Під час тестування ці квоти були перевищені, що призвело до помилок SSL з'єднання. Для вирішення цієї проблеми було необхідно очікувати отримання квот.

Розглянемо помилки доступу ПК до хмарних ресурсів. Для забезпечення доступу до API ПК було використано Azure Application Routing. Використання цього підходу вимагає налаштування доступу кластера до Azure DNS для керування DNS записами. Під час налаштування було допущено помилки, які зробили неможливим доступ кластера до створення необхідних DNS записів. Це зробило недоступними ресурси, запущені в кластері.

Для зберігання даних програмою, було використано Azure Blob Storage [21]. Для доступу до сховища було використано підхід "Workload Identity". Використання цього підходу вимагає створення в кластері ServiceAccount з оголошеною при створенні Workload Identity назвою у відповідному Namespace. Конфігурація Workload Identity не відповідала вимогам сховища, що спричинило тимчасову недоступність до збережених даних.

Для оптимального використання ресурсів було необхідно визначити необхідні параметри та структуру апаратного забезпечення. Ці параметри включали: кількість нод в кластері; кількість ядер процесора та параметри цих ядер; архітектуру процесора; кількість оперативної пам'яті; розмір диска; тощо.

Під час вибору параметрів була проведена велика кількість тестів продуктивності та порівнянь вартостей різних конфігурацій. Деякі конфігурації призводили до значного зниження продуктивності, а деякі, хоч й при збільшенні продуктивності, призводили до значного збільшення вартості інфраструктури. В результаті тестування було обрано наступні параметри:

- Кількість нод в кластері: 1-5
- ОС: Linux
- архітектура ЦП: ARM64
- процесор: Ampere Altra
- кількість ядер: 32
- обсяг ОЗУ: 64 GB
- обсяг диску: 32 GB

Розглянемо виконане налаштування розміщення ресурсів. Azure дозволяє розміщувати ресурси в різних регіонах світу. Різні ресурси мають різну ціну в різних регіонах, тож правильний вибір регіону може суттєво вплинути на вартість інфраструктури.

Для багатьох різновидів використання регіону є важливим для розміщення апаратного забезпечення ближче до користувача, балансування навантаження, відповідності до вимог законів про зберігання даних, тощо. Для використання даного ПК ці аспекти не були важливі, тому було прийнято рішення зосередитись на порівнянні вартості. Після порівнянь був обраний регіон центральної Індії (CentralIndia).

Далі було визначено необхідні налаштування в програмі доступу до ресурсів в хмарному середовищі.

Розглянемо керування ресурсами в Kubernetes кластері. Для доступу до сховища Azure необхідно налаштувати авторизацію в хмарному середовищі та адресу сховища. Авторизація була реалізована за допомогою підходу "Workload Identity", який повністю реалізований в бібліотеці клієнтів, наданою Azure. Під час налаштування авторизації ніяких помилок не виникало. Початково зазначена адреса сховища виявилася некоректною, що унеможливило підключення до Azure Blob Storage. Помилку було виправлено шляхом оновлення конфігурації програми.

Програмне рішення було розроблено з використанням архітектури «головний сервіс – працюючий сервіс». Головний сервіс автоматично створює працюючі сервіси, які безпосередньо проводять розрахунки. Для авторизації під час створення Kubernetes pods працюючих сервісів також використовувався підхід "Workload Identity", тому нових помилок пов'язаних з авторизацією не виникло. У процесі створення Kubernetes pod спостерігались конфлікти з обмеженнями ресурсів, які були усунені шляхом зміни конфігурацій запитів на розгортання.

Розглянемо оновлення назв файлів для використання Azure Blob Storage [21]. Azure сховище дає можливість завантажувати в нього і вивантажувати з нього файли за допомогою інтерфейсу користувача та бібліотеки клієнта, за допомогою якої налаштовується програмна взаємодія зі сховищем. Тестування виявило невідповідність між поведінкою інтерфейсу користувача та клієнтської бібліотеки, яка не підтримувала файли з назвами, що містять кириличні символи. Для виправлення цієї помилки було оновлено імена всіх файлів з вхідними і вихідними даними так, щоб використовувалась тільки латиниця і були відсутні пробіли.

Наступним етапом була підготовка ПК до роботи в хмарному середовищі. Під час першого запуску ПК в хмарному середовищі було виявлено, що передача даних між вузлами, які займаються розрахунками, займає значну частину часу загальної роботи. Причиною виявленої затримки передачі даних було використання HTTP-протоколу, який, хоч і зручний для візуалізації, є менш ефективним для високошвидкісної обробки великих обсягів даних. Цей

підхід є зручним для читання даних людиною, але не завжди є ефективним з точки зору об'єму інформації та швидкості її обробки. Для виправлення даної помилки було використано безпосередньо TCP протокол і передавання даних, збережених в масив байтів. Це прискорило передачу даних між вузлами на 43%. Таким чином, отриманий виграш у продуктивності й масштабованості компенсує початкові витрати часу на налагодження. Зокрема, використання ARM-процесорів дозволило суттєво знизити вартість інфраструктури, а перехід на TCP протокол скоротити затримки у взаємодії компонентів системи.

За замовчуванням Java програма має обмеження для максимально доступного об'єму оперативної пам'яті, що складає 25% від її загального об'єму [22]. В випадку, що аналізується, коли виконання ПК запущено в хмарному середовищі, воно є єдиним процесом, який використовує доступні ресурси. На підібраних налаштуваннях апаратного забезпечення 25% – це 16Gb, тобто 48Gb оперативної пам'яті не використовується взагалі. Для використання всієї доступної пам'яті було вказано додаткові аргументи в команді для запуску програми.

Виявлено, що стандартне обмеження на використання оперативної пам'яті в Java призводить до появи помилки OutOfMemory при виконанні операцій на великих матрицях. Це стається на апаратному забезпеченні з менш ніж 32Gb оперативної пам'яті і тільки під час обчислень на матрицях великих розмірностей. Для виправлення цієї помилки достатньо змінити максимальне обмеження на використання оперативної пам'яті.

Розроблений та реалізований ПК підтримує багатопоточний режим виконання. Під час розробки була проставлена фіксована кількість потоків, що є оптимальною для персонального комп'ютера, на якому виконувалась розробка. Хмарне середовище дає можливість запускати програму в середовищах з різними конфігураціями. Встановлення фіксованої кількості потоків, яка була оптимальною для локального середовища, виявилося недоцільним для масштабованого хмарного середовища, що призвело до зниження ефективності. Для уникнення неефективної поведінки програми було автоматизовано визначення кількості потоків на основі характеристик системи.

Стисло опишемо підготовку ПК до запуску на ARM процесорі. Віртуальні машини на ARM процесорах є значно дешевшими в хмарному середовищі Azure. Для оптимізації витрат було оновлено налаштування збирання Docker image (образу) програми для підтримки ARM процесорів.

Зібрати програму, яка підтримує роботу на ARM процесорі, є можливим тільки в середовищі, яке працює на ARM процесорі. Для збирання образу було використано GitHub Actions, який дає можливість безкоштовно запускати процеси для збирання і тестування програм в середовищах на різних типах процесорів включаючи, ARM архітектуру.

Описаний процес налаштування та тестування ПК у хмарному середовищі тривав 44 робочі години. На рис. 1 представлено залежність кількості виявлених помилок N_i, що фіксувалися протягом кожного інтервалу тестування тривалістю 4 години, від часу. По горизонтальній осі відкладається накопичений час тестування t, по вертикальній – кількість помилок, виявлених у відповідному інтервалі.

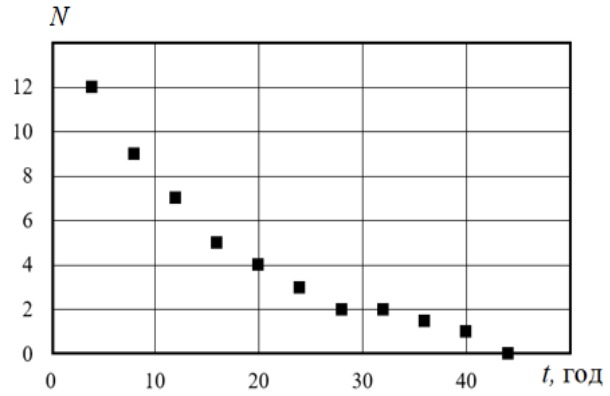


Рис. 1 – Кількість виявлених помилок N_i у кожному 4-годинному інтервалі впродовж загального часу тестування t

Постановка задачі та визначення надійності розгорнутого програмного забезпечення.

Одним з нових ефективних методів побудови математичних моделей для опису та прогнозування надійності програмних засобів є використання підходів із застосуванням диференційних рівнянь [9]. Як правило, використовуються лінійні диференційні рівняння першого порядку, що лінійно пов'язують швидкість поточної накопиченої кількості помилок N з самою даною величиною з коефіцієнтом, що є інтенсивністю програмних збоїв.

В даній роботі для оцінювання надійності програмного засобу запропоновано застосування диференційного рівняння так званого автомодельного типу, яке ефективно використовується при моделюванні довговічності, а отже й надійності, елементів машин та споруд [23]. Конкретний вигляд застосованого автомодельного рівняння відповідає еволюційному рівнянню для параметру суцільності, запропонованому у роботі [24], яке й використовується для аналізу довговічності.

Розглянемо диференційне рівняння першого порядку з нелінійною залежністю поточної накопиченої кількості помилок N. Параметр $\psi(t)$, що використовується при аналізі, відповідатиме ймовірності виникнення відмов (помилок) у час t, та отримає назву параметру відмов. Диференційне рівняння для параметру відмов $(0 < \psi \le 1)$ запишемо у наступному вигляді:

$$\frac{d\psi}{dt} = D\left(\frac{N}{\psi}\right)^m, \ \psi(0)=1, \ \psi(t_r)=0. \qquad (1)$$

Тут D та m є константами, що необхідно

визначити з експериментальних даних щодо тестування програмного засобу; t_r – час досягнення потрібної надійності з ймовірністю (0.98-0.99). У застосованих початкових умовах при t=0: $\psi(0)$=1, тобто ймовірність відмов дорівнює 1. Як видно, інтегрування рівняння (1) не є можливим проводити до досягнення значення $\psi(t$=0, але це не є проблемою при прогнозуванні, залишаючи 1-2% на невраховані фактори.

Інтегрування рівняння (1) з заданою початковою умовою надає залежність параметру відмов від часу:

$$\psi = \left[1 + D(m+1)N^m t\right]^{\frac{1}{m+1}} \qquad (2)$$

Загальну кількість помилок N_*, що виникає при $t=t_r$ є можливим визначити при реалізованому значенні $\psi(t_r)$=0:

$$N_* = \left[-\frac{1}{(m+1)Dt_r}\right]^{\frac{1}{m}}, \qquad (3)$$

як й саме значення часу забезпечення надійності програмного засобу:

$$t_r = -\frac{1}{D(m+1)N_*^m}. \qquad (4)$$

Запропонований підхід було використано для аналізу надійності роботи програмного засобу, що є подібним до описаного розгорнутого у хмарному середовищі ПК. Експериментально отриману залежність числа відмов на кожному інтервалі часу, представлену на рис. 1, було перебудовано для опису повного числа отриманих помилок всіх типів. Її представлено на рис. 2. Результат перебудови даної залежності в напівлогарифмічній системі координат представлено на рис. 3.

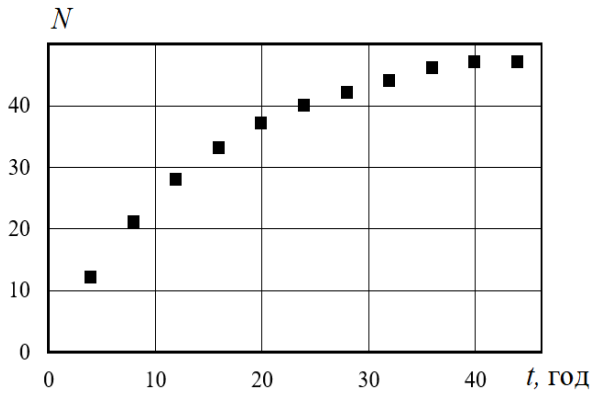
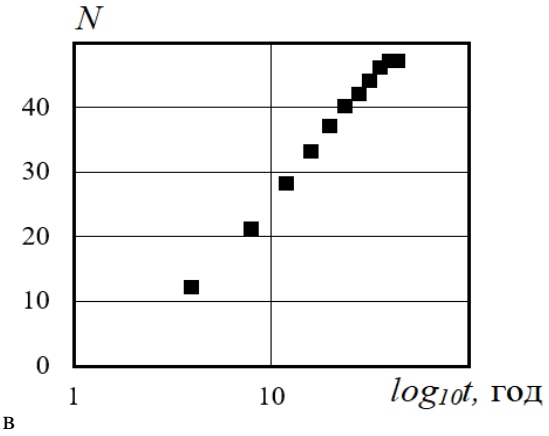


Рис. 2 – Залежність загального числа помилок N від часу



в

Рис 3 – Залежність загального числа помилок N від часу. Напівлогарифмічна шкала

Як видно з аналізу кривої на рис. 3, вона на практично кожному діапазоні часу має лінійний характер. Це дозволяє використати для її опису степеневу залежність від N у рівнянні (1). Для визначення значень констант, що входять до цього рівняння, скористаємось даними випробувань при тестуванні при початкових значеннях часу 4 та 8 год. Розв'язання системи алгебраїчних рівнянь типу (4) для двох пар значень (N_i, t_i), i=1,2 надає значення констант: D= 11.37 год$^{-1}$, m=-1.24.

Отримані параметри моделі D та m були визначені на основі емпіричних даних, визначених у ході тестування розглянутого програмного комплексу. Використання цих значень констант при інтегруванні рівняння (1) надає можливість отримати залежність параметру відмов ψ від часу, яку для розглянутого процесу тестування ПК для роботи у хмарному середовищі представлено на рис. 4. Як видно з наведеного графіку, автомодельне рівняння (1) цілком задовільно описує процес забезпечення надійності системи з мінімальними відхиленнями на рівні 5-7 % на останніх етапах тестування. Зауважимо, що певною мірою таке відхилення є корисним у зв'язку з тим, що при будь-якому процесі прогнозування завжди можуть лишитись невраховані в аналізі фактори.

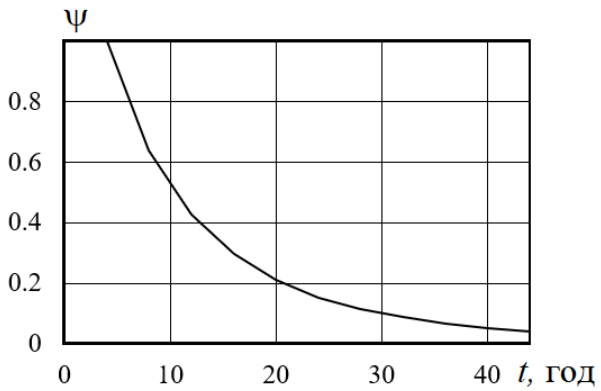


Рис. 4 – Залежність параметру відмов ψ від часу

Слід зауважити, що побудована модель базується на даних одного повного процесу налагодження та тестування програмного комплексу тривалістю 44 години. У класичному розумінні надійність та ймовірність відмов є статистичними характеристиками, що вимагають багаторазового експериментального підтвердження. Проте в умовах сучасних CI/CD-процесів та інженерного налагодження складних обчислювальних програм в хмарному середовищі часто відсутня можливість провести велику кількість незалежних повторів. У таких випадках доцільним є використання методів, що дозволяють здійснювати прогнозування на основі поточних характеристик процесу виявлення відмов [9].

Застосований у цій роботі підхід розглядає надійність як функцію часу та накопиченої кількості помилок і дозволяє з достатньою точністю прогнозувати момент досягнення цільового рівня надійності. Такий підхід до прогнозування надійності програм надає можливість оцінити кількість потрібного часу та числа можливих помилок при тестуванні. Для ідентифікації моделі є необхідним знання щодо процесу тестування лише на його початковому етапі. Така інформація може бути потрібною для визначення витрат для загального доведення системи. Також у деяких випадках є можливим використання даної математичної моделі й без попередньої інформації про тестування, наприклад на етапі проєктування. Для цього є необхідним порівняння метрик (наприклад, мір Холстеда [9]) даного розглянутого ПК та того, що розробляється. При більшій складності програмної системи при прогнозуванні можливо передбачити більший час тестування та більшу кількість помилок, що безперечно, потребує подальшої перевірки.

Практичне значення використання побудованої моделі полягає в тому, що вона надає можливість кількісно оцінити момент досягнення бажаного рівня надійності, що є критично важливим при розгортанні програмних комплексів у хмарному середовищі. Такий прогноз дозволяє оптимізувати тривалість тестування, мінімізувати витрати на використання обчислювальних ресурсів і зменшити загальний час виходу продукту на етап експлуатації. Модель також може використовуватись як частина автоматизованої процедури прийняття рішення щодо завершення налагодження, або як критерій ефективності альтернативних конфігурацій середовища. Врахування помилок, спричинених як технічними обмеженнями, так і конфігураційними чинниками, забезпечує системний підхід до прогнозування, не зосереджуючись на суб'єктивному "людському факторі", а лише на об’єктивно зафіксованих наслідках.

Висновки. У статті надано опис процесу тестування та налагодження програмного забезпечення на основі програмного комплексу для реалізації скінченноелементного аналізу у хмарному середовищі. Аналізуються типи та причини виникнення помилок при налаштуванні та виконанні програмного коду. За аналізом залежності числа отриманих помилок від часу запропоновано математичну модель для прогнозування надійності роботи аналогічних програмних засобів. Модель побудовано на базі інтегрування за часом диференційного рівняння автомодельного типу для параметру відмов, близькість якого до нуля у певний інтервал часу й забезпечує прогноз потрібної надійності програмного засобу. Показано задовільну здатність до прогнозування надійності програмних засобів розглянутої конфігурації при порівнянні з реальними даними тестування. Показник параметра відмов, що розраховується в рамках моделі, може бути використаний у практиці як критерій завершення етапу тестування, а також як метрика ефективності обраної архітектури або конфігурації середовища.

Запропонована модель може бути адаптована для програмних систем, що функціонують в обчислювальних середовищах з нестаціонарними характеристиками, та яким притаманні процеси накопичення та усунення помилок різного типу у часі. Це відкриває перспективу її використання в інших практичних задачах, пов’язаних із забезпеченням цільової надійності в умовах обмежених ресурсів.

Список літератури

1. *Sehgal N. K.* Cloud computing / *N. K. Sehgal*, *P. Ch. Bhatt.* – Heidelberg: Springer, 2018. https://doi.org/10.1007/978-3-319-77839-6
2. *Rittinghouse J. W.* Cloud computing: implementation, management, and security / *J. W. Rittinghouse*, *J. F. Ransome*. – Boca Raton : CRC Press, 2017. https://doi.org/10.1201/9781439806814
3. *Marinescu D. C.* Cloud computing: theory and practice / *D. C. Marinescu*. – Amsterdam : Morgan Kaufmann, 2022. – 61 p.
4. *Alam T.* Cloud Computing and its role in the Information Technology / *T. Alam* // IAIC Transactions on Sustainable Digital Innovation (ITSDI). – 2021. – V. 1, № 2. – P. 108–115. https://doi.org/10.34306/itsdi.v1i2.103
5. *Alzakholi O.* Comparison among cloud technologies and cloud performance / *O. Alzakholi,* [et al.] // Journal of Applied Science and Technology Trends. – 2020. – V. 1, № 1. – P. 40–47. https://doi.org/10.38094/jastt1219
6. Mathematics in Software Reliability and Quality Assurance / eds. T. *Dohi*, *S. Liu*. – Basel : MDPI, 2022. – 218 p.
7. *Lyu M. R.* Software reliability theory / *M. R. Lyu* // Encyclopedia of Software Engineering. – 2002. – V. 2. – P. 1611–1630. https://doi.org/10.1002/0471028959.sof329
8. *Lyu M.* Software reliability engineering: A roadmap / *M. Lyu* // Future of Software Engineering (FOSE’07). – 2007. – P. 153–170. https://doi.org/10.1109/FOSE.2007.24
9. *Yamada S.* Software reliability modeling: fundamentals and applications / *S. Yamada*. – Tokyo : Springer, 2014. – V. 5. – 90 p. https://doi.org/10.1007/978-4-431-54565-1
10. *Zhu Q.-H.* Task scheduling for multi-cloud computing subject to security and reliability constraints / *Q.-H. Zhu*, [et al.] // IEEE/CAA Journal of Automatica Sinica. – 2021. – V. 8, № 4. – P. 848–865. https://doi.org/10.1109/JAS.2021.1003934
11. *Sefati S.* Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation. / *Sefati S.*, *M. Mousavinasab*, *R. Zareh Farkhady*. // The Journal of Supercomputing. – 2022. – V. 78, № 1. – P. 18–42. https://doi.org/10.1007/s11227-021-03810-8
12. *Yanamala A. K.* Emerging challenges in cloud computing security: A comprehensive review / *A. K. Yanamala* // International Journal of Advanced Engineering Technologies and Innovations. – 2024. –

V. 1, № 4. – P. 448–479.
13. *Parast F. Kh*. Cloud computing security: A survey of service-based models / *F. Kh. Parast*, [et al.] // Computers & Security. – 2022. – V. 114. – Article ID: 102580. https://doi.org/10.1016/j.cose.2021.102580
14. *Maciel P*. A survey on reliability and availability modeling of edge, fog, and cloud computing / *P. Maciel*, [et al.] // Journal of Reliable Intelligent Environments. – 2022. – P. 1–19.
15. *Бреславський Д. В.* Проектування та розробка скінченноелементного програмного забезпечення / *Д. В. Бреславський, Ю. М. Коритко, О. А. Татарінова.* – Харків : Підручник НТУ «ХПІ», 2017. – 232 с.
16. *Бреславський Д.* Скінченноелементне програмне забезпечення для розв'язання тривимірних задач з використанням хмарних технологій / *Д. Бреславський*, *М. Бородін*, *О. Татарінова*, *А. Сенько* // Вісник Національного технічного університету «ХПІ». Серія: Динаміка та міцність машин. – 2024. – № 2. – С. 23–29.
17. *Buchanan S.* Introducing Azure Kubernetes Service / *S. Buchanan*, J. *Rangama, N. Bellavance*. – 2019. https://doi.org/10.1007/978-1-4842-5519-3
18. *Collier M.* Fundamentals of Azure / *M. Collier*, *R. Shahan*. – Redmond : Microsoft Press, 2016.
19. *Shirinkin K*. Getting Started with Terraform / *K. Shirinkin*. – Birmingham : Packt Publishing Ltd, 2017.
20. *Saroar S. G.* Developers' perception of github actions: A survey analysis / *S. G. Saroar*, *M. Nayebi* // Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering. – 2023. – P. 121–130. https://doi.org/10.1145/3593434.3593475
21. *Daher Z.* Cloud storage comparative analysis: Amazon Simple Storage vs. Microsoft Azure Blob Storage. / *Z. Daher*, *H. Hajjdiab* // International Journal of Machine Learning and Computing. – 2018. – V. 8, № 1. – P. 85–89. https://doi.org/10.18178/ijmlc.2018.8.1.668
22. Memory Management // Oracle [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/en/graalvm/jdk/21/docs/reference-manual/native-image/optimizations-and-performance/MemoryManagement/#overview-1
23. *Bolotin V. V.* Mechanics of fatigue / *V. V. Bolotin*. – Boca Raton : CRC Press, 2020. – 480 p. https://doi.org/10.1201/9780138747848
24. *Kachanov L. M.* Rupture time under creep conditions / *L. M. Kachanov* // International Journal of Fracture. – 1999. – V. 97, № 1. – P. 11–18. https://doi.org/10.1023/A:1018671022008

References (transliterated)

1. Sehgal, N. K., and Bhatt, P. Ch. *Cloud Computing*. Springer, 2018. https://doi.org/10.1007/978-3-319-77839-6
2. Rittinghouse, J. W., and Ransome, J. F. *Cloud Computing: Implementation, Management, and Security*. CRC Press, 2017. https://doi.org/10.1201/9781439806814
3. Marinescu, D. C. *Cloud Computing: Theory and Practice*. Morgan Kaufmann, 2022.
4. Alam, T. Cloud Computing and Its Role in the Information Technology. *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, vol. 1, no. 2, 2021, pp. 108–115. https://doi.org/10.34306/itsdi.v1i2.103
5. Alzakholi, r O., et al. Comparison among Cloud Technologies and Cloud Performance. *Journal of Applied Science and Technology Trends*, vol. 1, no. 1, 2020, pp. 40–47. https://doi.org/10.38094/jastt1219
6. Dohi, T., and Liu, S. *Mathematics in Software Reliability and Quality Assurance*. MDPI, 2022.
7. Lyu, M. R. Software Reliability Theory. *Encyclopedia of Software Engineering*, vol. 2, 2002, pp. 1611–1630. https://doi.org/10.1002/0471028959.sof329
8. Lyu, M. Software Reliability Engineering: A Roadmap. *Future of Software Engineering (FOSE'07)*, IEEE, 2007, pp. 153–170. https://doi.org/10.1109/FOSE.2007.24
9. Yamada, S. *Software Reliability Modeling: Fundamentals and Applications*. vol. 5, Springer, 2014. https://doi.org/10.1007/978-4-431-54565-1
10. Zhu, Q.-H., et al. Task Scheduling for Multi-Cloud Computing Subject to Security and Reliability Constraints. *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 4, 2021, pp. 848–865. https://doi.org/10.1109/JAS.2021.1003934
11. Sefati, S., Mousavinasab, M., and Zareh Farkhady, R. Load Balancing in Cloud Computing Environment Using the Grey Wolf Optimization Algorithm Based on the Reliability: Performance Evaluation. *The Journal of Supercomputing*, vol. 78, no. 1, 2022, pp. 18–42. https://doi.org/10.1007/s11227-021-03810-8
12. Yanamala, A. K. Emerging Challenges in Cloud Computing Security: A Comprehensive Review. *International Journal of Advanced Engineering Technologies and Innovations*, vol. 1, no. 4, 2024, pp. 448–479.
13. Parast, F. Kh., et al. Cloud Computing Security: A Survey of Service-Based Models. *Computers & Security*, vol. 114, 2022, article ID 102580. https://doi.org/10.1016/j.cose.2021.102580
14. Maciel, P., et al. A Survey on Reliability and Availability Modeling of Edge, Fog, and Cloud Computing. *Journal of Reliable Intelligent Environments*, 2022, pp. 1–19.
15. Breslavskyi, D. V., Korytko, Yu. M., and Tatarinova, O. A. *Proektuvannia ta rozrobka skinchennoelementnoho prohramnoho zabezpechennia* [Design and Development of Finite Element Software]. Pidruchnyk NTU "KhPI", 2017.
16. Breslavskyi, D. V., Borodin, M., Tatarinova, O., and Senko, A. Skinchennoelementne prohramne zabezpechennia dlia rozviazannia tryvymirnykh zadach z vykorystanniam khmarnykh tekhnolohii [Finite Element Software for Solving 3D Problems Using Cloud Technologies]. *Visnyk Natsionalnoho Tekhnichnoho Universytetu "KhPI". Seriia: Dynamika ta mitsnist mashyn*, no. 2, 2024, pp. 23–29.
17. Buchanan, S., Rangama, J., and Bellavance, N. Introducing Azure Kubernetes Service. 2019. https://doi.org/10.1007/978-1-4842-5519-3
18. Collier, M., and Shahan, R. *Fundamentals of Azure*. Microsoft Press, 2016.
19. Shirinkin, K. *Getting Started with Terraform*. Packt Publishing, 2017.
20. Saroar, S. G., and Nayebi, M. Developers' Perception of GitHub Actions: A Survey Analysis. *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, 2023, pp. 121–130. https://doi.org/10.1145/3593434.3593475
21. Daher, Z., and Hajjdiab, H. Cloud Storage Comparative Analysis: Amazon Simple Storage vs. Microsoft Azure Blob Storage. *International Journal of Machine Learning and Computing*, vol. 8, no. 1, 2018, pp. 85–89. https://doi.org/10.18178/ijmlc.2018.8.1.668
22. *Memory Management*. Oracle, https://docs.oracle.com/en/graalvm/jdk/21/docs/reference-manual/native-image/optimizations-and-performance/MemoryManagement/#overview-1. Accessed 1 Aug. 2025.
23. Bolotin, V. V. *Mechanics of Fatigue*. CRC Press, 2020. https://doi.org/10.1201/9780138747848
24. Kachanov, L. M. Rupture Time under Creep Conditions.*International Journal of Fracture*, vol. 97, no. 1, 1999, pp. 11–18. https://doi.org/10.1023/A:1018671022008

Надійшла (received) 12.06.2025
Прийнята до друку (accepted) 01.09.2025
Опублікована (published) 03.09.2025

Відомості про авторів/ About the Authors

Бреславський Дмитро Васильович (Breslavsky Dmytro)– доктор технічних наук, професор, Національний технічний університет «Харківський політехнічний інститут», професор кафедри комп'ютерного моделювання процесів та систем; м. Харків, Україна; тел.: (057)7076454; ORCID: https://orcid.org/0000-0002-3792-5504; email: Dmytro.Breslavsky@khpi.edu.ua

Бородін Марія Анатоліївна (Borodin Mariia)– Національний технічний університет «Харківський політехнічний інститут», аспірантка кафедри комп'ютерного моделювання процесів та систем; м. Харків, Україна; тел.: (057)7076454; ORCID: https://orcid.org/0009-0003-4479-7103; email: Mariia.Borodin@khpi.edu.ua

Грошевий Михайло Олександрович (Hroshevyi Mykhailo) – Національний технічний університет «Харківський політехнічний інститут», аспірант кафедри комп'ютерного моделювання процесів та систем; м. Харків, Україна;; тел.: (057)-707-64-54; ORCID: https://orcid.org/0009-0001-3156-4402; email: Mykhailo.Hroshevyi@khpi.edu.ua.

Татарінова Оксана Андріївна (Tatarinova Oksana) – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», завідувачка кафедри комп'ютерного моделювання процесів та систем; м. Харків, Україна; тел.: (057)-707-64-54; ORCID: https://orcid.org/0000-0003-3090-8469; e-mail: Oksana.Tatarinova@khpi.edu.ua