ISSN 2707-1898 (print)

Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2024.01.001
Article received 18.04.2024 p.
Article accepted 30.04.2024 p.
UDK 004.05

Correspondence author
M. M. Seniv
maksym.m.seniv@lpnu.ua

УДК 004.05

М. М. Сенів
Національний університет "Львівська політехніка", м. Львів, Україна

МЕТОД АВТОМАТИЗАЦІЇ ПРОЦЕСУ РОЗРАХУНКУ ПОКАЗНИКІВ НАДІЙНОСТІ ПРОГРАМНИХ СИСТЕМ ТА ЇХ КОМПОНЕНТІВ

Проаналізовано різні засоби розрахунку показників надійності програмних систем. Встановлено, що для визначення показників надійності програмних систем доцільно використовувати структурно-логічний аналіз блок-схем надійності, оскільки він наочно та найадекватніше відображає розрахунок показників надійності програмної системи загалом та її компонентів зокрема. Незважаючи на зовнішню простоту такого аналізу, його виконання не є тривіальним завданням, адже навіть сформулювати умову працездатності технічної системи непросто, особливо за наявності багатьох елементів із різноплановими зв'язками між ними, а якщо робити це вручну, дуже велика ймовірність людської помилки. Побудова та візуалізація графа станів / переходів є також нетривіальним завданням, оскільки кількість можливих станів програмної системи залежно від кількості елементів зростає експоненційно, та, своєю чергою, збільшує складність системи диференціальних рівнянь, розв'язання якої дає змогу розрахувати необхідні показники надійності. Визначено, що надійнісне проєктування складних програмних систем загалом, та їх компонентів зокрема, потребує автоматизації усіх його етапів, починаючи від складання блок-схеми надійності (reliability block diagram, RBD) і закінчуючи візуалізацією отриманих результатів. Розроблено метод автоматизації процесу розрахунку показників надійності програмних систем та їх компонентів, який складається із восьми кроків, та, на відміну від відомих, дає проєктанту змогу інтуїтивно зрозуміло вводити не тільки вхідні дані про структуру, а й саму архітектуру ПЗ із погляду його надійності, а також автоматизує всі етапи розрахунку показників надійності, від етапу побудови блок-схеми надійності до етапу виявлення розподілу ймовірностей перебування програмної системи у всіх можливих станах. Запропонований метод дає змогу використовувати в різних комбінаціях методи, алгоритми та програмні засоби, які використовують для надійнісного проєктування саме програмних систем, та вибрати із них найадекватніші потребам користувача в конкретній ситуації. Використання розробленого методу дає можливість зменшити вплив людського фактора та імовірності внесення помилки під час розрахунку показників надійності програмних систем на всіх етапах надійнісного проєктування та зменшити час його виконання як мінімум на 21 %.

Ключові слова: програмне забезпечення, блок-схема надійності, надійнісне проєктування, граф станів та переходів.

Вступ / Introduction

Згідно зі стандартом ISO/IEC 25010:2011 (System and software quality model) [1] надійність програмного забезпечення є однією із важливих характеристик його якості. Відповідно, розрахунок показників надійності програмних систем є актуальною проблемою, оскільки ступінь надійності програмної системи впливає на загальне функціонування та поведінку системи, від неї можуть залежати безпека та життя людей, особливо у разі використання систем спеціального або критичного призначення.

Для розрахунку показників надійності програмно-апаратних систем розроблено низку методів [2], [3], [4] та засобів [5], [6], [7], [8], але поки що не існує універсального методу розрахунку надійнісних показників програмного забезпечення, що пов'язано із динамікою зміни складності програмних систем, їх багатофункціональністю. До того ж вони не враховують повною мірою особливості саме програмного компонента складних програмно-апаратних комплексів. Ще одним аспектом цього завдання є необхідність автоматизації цього процесу, оскільки виконання робіт на всіх етапах надійнісного проєктування у ручному режимі може призвести до помилок та необґрунтованих часових затрат. Відповідно, існує потреба в розробленні нових та удосконаленні наявних методів і засобів автоматизації розрахунку показників надійності програмних систем та їх компонентів.

Об'єкт дослідження – процес підвищення надійності програмного забезпечення.

Предмет дослідження – методи і засоби автоматизації процесу розрахунку показників надійності програмних систем та їх компонентів.

Мета роботи – розроблення методу автоматизації процесу розрахунку показників надійності програмних систем та їх компонентів і опрацювання результатів надійнісного проєктування програмних систем довільного рівня складності.

Для досягнення зазначеної мети необхідно вирішити такі *основні завдання дослідження*:

- проаналізувати розвиток методів розрахунку надійнісних показників технічних систем;
- проаналізувати наявні засоби розрахунку показників надійності програмних систем;
- сформулювати вимоги до розроблюваного методу;
- розробити метод автоматизації процесу розрахунку показників надійності програмних систем та їх компонентів;
- верифікувати результати роботи розробленого методу.

Аналіз останніх досліджень та публікацій. У стандарті ISO/IEC 25010:2011 наведено такі визначення: якість програмної системи – це ступінь, в якому система задовольняє заявлені та неявні потреби різних зацікавлених сторін і, таким чином, забезпечує цінність; надійність (є характеристикою якості) – це ступінь, до якого система, продукт або компонент виконують задані функції за певних умов протягом певного проміжку часу [1]. Відповідно до моделі якості програмного забезпечення надійність складається із таких підхарактеристик [1]:

- готовність (availability) – ступінь, до якого система, продукт або компонент працюють і доступні, коли це необхідно для використання;
- зрілість (maturity) – ступінь, до якого система, продукт або компонент відповідають потребам у надійності за умови нормальної експлуатації;
- відмовостійкість (fault tolerance) – ступінь, до якого система, продукт або компонент працюють належно, попри наявні апаратні чи програмні збої;
- відновлюваність (recoverability) – ступінь, до якого у разі перерви або збою продукт або система можуть відновити дані, на які безпосередньо впливає, і відновити потрібний стан системи.

Для визначення цих підхарактеристик використовують різні критерії та показники, такі як: середній час повідомлення про несправність (mean fault notification time), середній час між відмовами (mean time between failure), середній час простою (mean down time) тощо [9]. Одним із класичних підходів до розрахунку цих та інших показників надійності є структурно-логічний аналіз технічних систем. За такого підходу для розрахунків параметрів надійності використовують блок-схеми надійності (англ. reliability block diagram, RBD), які графічно відображають взаємозв'язок елементів та їх вплив на працездатність системи загалом [2], [10], [11], [12]. За такого підходу умову працездатності системи можна описувати за допомогою функцій алгебри логіки, далі сформувати матрицю станів, де міститиметься інформація про функціонування системи з погляду її надійності, та візуалізувати результати у вигляді графа станів / переходів. Після цього формують систему диференціальних рівнянь Колмогорова – Чепмена, які дають змогу інтерпретувати надійнісну поведінку системи в часі як випадковий процес зі скінченною множиною значень і неперервною зміною аргументу (часу), тобто як дискретно-неперервний стохастичний процес. Розв'язок такої системи рівнянь дає змогу визначити розподіл ймовірностей безвідмовної роботи у всіх станах системи [2], [10].

Також, з метою поліпшення якості та підвищення надійності програмного забезпечення, застосовують методи прогнозування дефектності програмного забезпечення для виявлення потенційних помилок [13], [14], [15], [16]. Більшість програмних дефектів виникають на етапі розроблення програмного забезпечення у вихідному коді програми і спричинені низкою факторів, які виникають на різних етапах життєвого циклу продукту, а саме: помилки у програмному коді, проблеми комунікації, неточності вимог до програмного забезпечення, недостатня документація, складність системи, терміни виконання проєкту, людський фактор тощо [17], [18], [19], [20]. Є два основні класи методів класифікації в прогнозуванні дефектності програмного забезпечення: статистичний підхід [21], [22], [23], [24] (використовують традиційні статистичні моделі, такі як, наприклад, регресійні моделі) і підхід на основі машинного навчання [25], [26], [27], [28]. Разом з їхніми перевагами, ці методи мають певні недоліки, такі як неможливість застосування на ранніх етапах життєвого циклу програмного забезпечення, великі обсяги даних для використання методів машинного навчання тощо.

Відповідно, для визначення показників надійності програмних систем доцільно використовувати структурно-логічний аналіз блок-схем надійності, оскільки він наочно та найадекватніше відображає процес розрахунку показників надійності програмної системи загалом та її компонентів зокрема. Незважаючи на зовнішню простоту такого аналізу, його виконання не є тривіальним завданням, адже навіть сформулювати умову працездатності технічної системи непросто, особливо за наявності багатьох елементів із різноплановими зв'язками між ними, а якщо робити це вручну, ймовірність людської помилки дуже велика. Побудова та візуалізація графа станів / переходів є також нетривіальним завданням, оскільки кількість можливих станів програмної системи залежно від кількості елементів зростає експоненційно, та, своєю чергою збільшує складність системи диференціальних рівнянь. Отже, надійнісне проєктування складних програмних систем загалом, та їх компонентів зокрема, потребує автоматизації усіх його етапів, починаючи від складання блок-схеми надійності (RBD) і закінчуючи візуалізацією отриманих результатів, відповідно, актуальною є автоматизація процесу розрахунку показників надійності програмних систем та їх компонентів і опрацювання результатів надійнісного проєктування програмних систем довільного рівня складності.

Результати дослідження та їх обговорення / Research results and their discussion

Опис методу автоматизації процесу розрахунку показників надійності програмних систем та їх компонентів. Для вирішення завдання автоматизації процесу надійнісного проєктування складних програмних систем розроблено метод автоматизації процесу розра-

хунку показників надійності програмних систем та їх компонентів, який складається із восьми кроків та графічно відображений на рисунку.

Крок 1. Подання структури програмної системи з погляду її надійності. Структура програмної системи з погляду її надійності повинна бути подана у вигляді блок-схеми надійності (англ. reliability block diagram RBD), у вигляді сукупності однофункціональних вузлів (модулів) та необхідних взаємозв'язків між ними. Блок-схеми надійності широко використовують для подання структури апаратних та програмно-апаратних систем з погляду їх надійності. Водночас, представлення програмної системи в такому вигляді має певну специфіку, оскільки інтенсивність відмов у момент часу $\lambda(t)$ кожного окремого елемента цієї схеми в апаратних системах задає виробник, виконавши випробування партії таких елементів. У програмних системах поширенішою є практика визначення кількості відмов на певному інтервалі тестування. Тут доцільним буде використання відповідних моделей надійності ПЗ, які дадуть змогу визначити інтенсивність відмов у момент часу $\lambda(t)$ кожного окремого елемента програмної системи на підставі даних тестування. Щодо подання загальної структури програмної системи у вигляді блок-схеми надійності використання відповідних програмних засобів [29] дає проєктанту змогу інтуїтивно зрозуміло вводити не тільки вхідні дані про структуру, а й саму архітектуру ПЗ із погляду його надійності. Вихідними даними для побудови блок-схеми надійності можуть слугувати UML-діаграми, описи у вигляді онтологій тощо.

Крок 2. Автоматизоване визначення умови працездатності. Якщо блок-схема надійності (або, інакше кажучи, структура системи з погляду її надійності) містить менш ніж десять елементів, то для визначення умови працездатності доцільно застосовувати рекурсивний алгоритм обходу схеми [30]. Хоча цей алгоритм накладає певні обмеження під час побудови структурної схеми надійності й непридатний для визначення умов працездатності для схем певного типу, оскільки не має інформації про повну топологію схеми, а лише будує умову працездатності під час обходу схеми, а також коректно працює, якщо в схемі чітко встановлені вузли початку і кінця для паралельних підсистем, його застосування зумовлено кращими часовими характеристиками в разі опрацювання схем з малою кількістю елементів [31].

Якщо в системі більше від десяти елементів, то для визначення умови працездатності доцільно застосувати метод автоматизованого визначення функції працездатності [31]. Використання цього методу зумовлено як його універсальністю під час застосування для аналізу топології схем зі складною структурою, так і кращими часовими характеристиками на структурних схемах великої розмірності (100 та більше елементів) [31]. На малих схемах (до десяти елементів) рекурсивний алгоритм виконує побудову умови працездатності швидше. Це зумовлено тим, що метод автоматизованого визначення функції працездатності ґрунтується на аналізі всієї схеми і вже після цього починає будувати умову працездатності, а рекурсивний алгоритм робить це у міру обходу схеми.

Крок 3. Формування матриці станів / переходів. У матриці станів міститься інформація про функціонування системи в сенсі її надійності. Кожен рядок матриці являє собою вектор, компонентами якого слугують ознаки того, в якому стані перебуває кожен елемент, коли сама система у стані *i*. Формування матриці станів / переходів відбувається на основі даних, отриманих на Кроці 1 та Кроці 2. Множина станів і переходів системи є вхідними даними для подальшого складання диференціальних рівнянь Колмогорова – Чепмена. Після формування матриці станів / переходів необхідно прийняти рішення про необхідність формування графа станів / переходів для візуалізації отриманих результатів. Якщо потрібно сформувати граф станів / пере ходів, переходимо до Кроку 4, якщо ні – до Кроку 6.

Крок 4. Формування *n*-арного графу станів / переходів. Через відмови і відновлення система в дискретні моменти часу переходить з одного стану в інший. Під час тривалої експлуатації вона може побувати в кожному із можливих станів неодноразово. Тоді зручно її функціонування описувати *n*-арним графом, вузлам якого приписують стани системи, а гілкам – можливі переходи зі стану в стан. Якщо в графі є *n* вузлів, то серед них буде *k*-вузлів, які відповідають несправним станам, і (*n*–*k*) – справним. На цьому кроці потрібно вирішити, чи доцільно формувати *n*-арний граф станів / переходів. Якщо так, то доцільно застосувати алгоритм побудови графа станів / переходів на основі імітації послідовних відмов елементів [32], після чого перейти до Кроку 6. Якщо ні – переходимо до Кроку 5.

Крок 5. Формування кругового графа станів / переходів. Як альтернативу *n*-арному графу станів / переходів з метою візуалізації результатів можна використовувати круговий граф. Якщо на цьому кроці прийнято рішення сформувати круговий граф, то необхідно застосувати алгоритм формування кругового графа [33] та перейти на Крок 6.

Крок 6. Автоматизоване формування системи диференціальних рівнянь Колмогорова – Чепмена. Надійнісну поведінку системи в часі можна інтерпретувати як випадковий процес зі скінченною множиною значень і неперервною зміною аргументу (часу), тобто як дискретно-неперервний стохастичний процес. Часові залежності безумовних ймовірностей значень випадкового процесу та умовних ймовірностей переходів описуються системою диференціальних рівнянь Колмогорова – Чепмена. Вхідними даними для побудови системи рівнянь є множина станів і переходів для досліджуваної програмної системи. На цьому кроці доцільно використати алгоритм автоматизованого оброблення системи диференціальних рівнянь Колмогорова – Чепмена [34], який дає змогу побудувати систему диференціальних рівнянь для опису функціонування програмної системи. Після формування системи диференціальних рівнянь Колмогорова – Чепмена переходимо до Кроку 7.

Крок 7. Вибір стандартного інструменту / методу автоматизованого розв'язування системи диференціальних рівнянь Колмогорова – Чепмена. На цьому кроці необхідно вибрати інструмент або метод для автоматизованого розв'язування системи диференціальних рівнянь. Якщо прийнято рішення застосовувати відомі програмні комплекси типу Matlab, Mathcad, Scilab Online та/або їхні API, то варто використати програмний засіб, описаний у [34], який використовує алгоритм, зазначений на Кроці 6, для формування системи

диференціальних рівнянь і розв'язує її за допомогою Matlab API. Якщо ж вирішили застосовувати інші інструменти та/або методи для автоматизованого розв'язування системи диференціальних рівнянь, то доцільно використати програмний засіб [35] розв'язування систем диференціальних рівнянь Колмогорова – Чепмена для автоматизації надійнісного проєктування, який дає змогу розв'язувати ці системи рівнянь методом Рунге – Кутта без залучення спеціалізованих програмних продуктів (Matlab, Mathcad). Він, за рахунок інтеграції в програмний комплекс автоматизації надійнісного проєктування, дає змогу в два – три рази швидше, ніж програмні додатки із використанням API-функцій спеціалізованих програмних продуктів (Matlab, Mathcad), опрацьовувати вхідні дані великих об'ємів, та візуалізувати результати обчислень. Використання вищезазначеного програмного засобу також знімає проблему купівлі ліцензій спеціалізованих програмних продуктів (Matlab, Mathcad) [7].

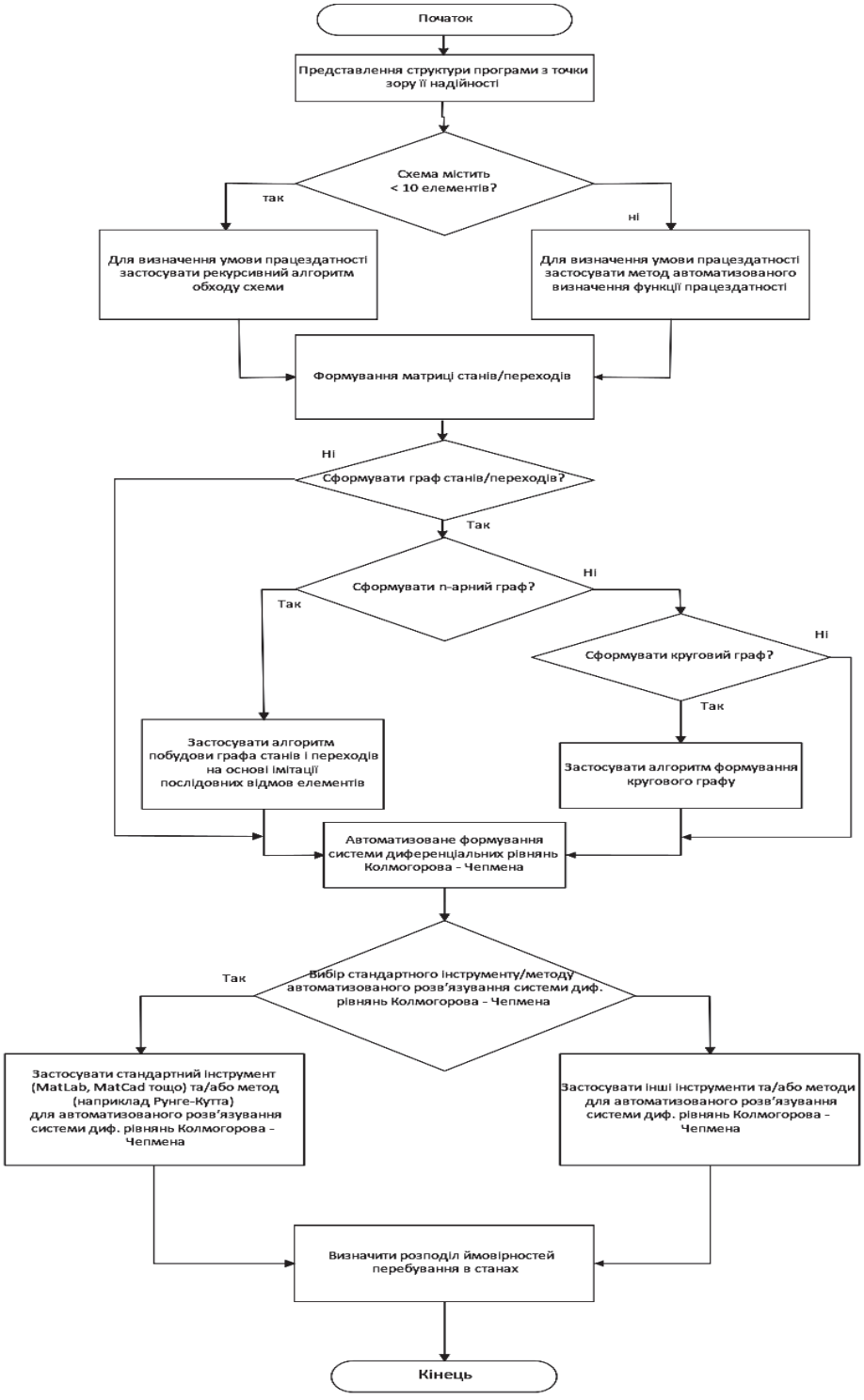

Рис. 1. Метод автоматизації розрахунку показників надійності програмних систем та їх компонентів /
A method of automating the process of calculating reliability indicators of software systems and their components

Крок 8. Визначити розподіл ймовірностей перебування у станах. Знайшовши розв'язки системи диференціальних рівнянь Колмогорова – Чепмена одним із вищезазначених способів, можна отримати розподіл ймовірностей перебування програмної системи у *i*-му стані $P_i(t)$ протягом часу t для усіх можливих станів програмної системи. Знайдені числові значення ймовірностей перебування у відповідних станах дають змогу надалі обчислити такі показники надійності: середній час безвідмовної роботи, середній час роботи між відмовами, функцію готовності тощо.

Верифікація отриманих результатів. З метою верифікації одержаних результатів можна розглянути застосування розроблених засобів [29–35] на різних кроках запропонованого методу.

Крок 1. Подання структури програмної системи з погляду її надійності. Застосування засобів візуалізації блок-схеми надійності [29] дає змогу проєктанту інтуїтивно зрозуміло вводити не тільки вхідні дані про структуру, а й про саму архітектуру ПЗ із погляду його надійності та, відповідно, зменшити вплив людського фактора й імовірності внесення помилки.

Крок 2. Автоматизоване визначення умови працездатності. У разі блок-схеми надійності із менш ніж десятьма модулями використання рекурсивного алгоритму дасть виграш у часі близько 21 %, а застосування методу автоматизованого визначення функції працездатності забезпечить виграш у часі близько 26 %, за умови проєктування програмної системи, яка складається зі 100 та більше модулів [31].

Крок 4. Формування *n*-арного графа станів / переходів. Застосування алгоритму побудови графа станів / переходів на основі імітації послідовних відмов елементів [32], порівняно з відомими, забезпечує виграш у часі близько 20 %. Це зумовлено тим, що запропонований алгоритм використовує послідовну імітацію відмови і відновлень елементів системи, на відміну від наявних, які виконують пошук вглиб (імітація відмов елементів для однієї гілки до моменту критичної відмови). Використання послідовного перебору дає змогу пришвидшити злиття станів, що дублюються, оскільки визначення станів відбувається порівнево і всі стани, які дублюються, перебувають на одному рівні. Під час пошуку вглиб цей процес є набагато складнішим, що спричиняє значні затрати в часі [32].

Крок 6. Автоматизоване формування системи диференціальних рівнянь Колмогорова – Чепмена. Для досягнення цієї мети можна використати програмний модуль [34], який створює та відображає систему диференціальних рівнянь Колмогорова – Чепмена та роз'язує її за допомогою програмного пакета Matlab із використанням відповідного API. В [34] зазначено, що час оброблення системи, яка складається із 50 модулів, не перевищує 2,2 с. Це прийнятний результат для моделювання поведінки надійності програмних систем зі складною структурою.

Крок 7. Вибір стандартного інструменту / методу автоматизованого розв'язування системи диференціальних рівнянь Колмогорова – Чепмена. Вибір засобів автоматизованого розв'язування системи диференціальних рівнянь може здійснюватись з урахуванням наявності / відсутності ліцензій спеціалізованих програмних продуктів (Matlab, Mathcad) тощо та/або вимог щодо швидкодії цих засобів. Наприклад, програмний модуль розв'язування систем диференціальних рівнянь Колмогорова – Чепмена [35] дає змогу розв'язувати системи таких рівнянь методом Рунге – Кутта без залучення спеціалізованих програмних продуктів, і, за рахунок інтеграції в програмний комплекс автоматизації надійнісного проєктування, дає змогу в два – три рази швидше, ніж програмні додатки із використанням API-функцій спеціалізованих програмних продуктів, опрацьовувати вхідні дані великих об'ємів та візуалізувати результати обчислень. Детальні дані щодо швидкодії різних засобів наведено в [35].

Обговорення результатів дослідження. Застосування методів та алгоритмів автоматизації надійнісного проєктування дає змогу проєктанту зменшити часові та, відповідно, фінансові затрати на етапах проєктування, розроблення та тестування програмної системи, зменшити ймовірність внесення помилок та мінімізувати вплив людського фактора. Упродовж останніх років питання автоматизації та підвищення ступеня адекватності надійнісного проєктування розглядалось в багатьох роботах вітчизняних [2], [4], [6] та зарубіжних [3], [10], [11] авторів, але більшість з них зосередили увагу на застосуванні вищезазначених засобів для розрахунку надійності програмно-апаратних або апаратних систем. Використання таких методів для розрахунку надійнісних показників саме програмних систем має певну специфіку, пов'язану із відмінностями апаратного і програмного забезпечення з погляду його надійності. Наприклад, закони фізичного зношення апаратних елементів технічної системи не можна застосовувати для аналогічних елементів програмної системи, показник інтенсивності відмов у момент часу $\lambda(t)$ кожного окремого елемента в апаратних системах задає виробник на основі випробувань партії таких елементів, а в програмних системах поширенішою є практика визначення $\lambda(t)$ кожного окремого елемента програмної системи на підставі даних тестування тощо. Це накладає певні обмеження на можливість застосування методів надійнісного проєктування, які широко застосовували для розрахунку надійності апаратних систем, під час розрахунку показників надійності програмного забезпечення. Запропонований метод дає можливість використовувати в різних комбінаціях методи, алгоритми та програмні засоби, які застосовують для надійнісного проєктування саме програмних систем, та вибрати з них найадекватніші потребам користувача в конкретній ситуації. Із вищенаведеного можна також зробити висновок, що сумарний виграш у часі в разі застосування розробленого методу становитиме мінімум 21 % (виконання умов, описаних на Кроці 2), а максимум – 146 % (виконання умов, описаних на Кроці 2, Кроці 4 та Кроці 7).

Отже, за результатами виконаної роботи можна визначити наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження: розроблено метод автоматизації процесу розрахунку показників надійності програмних систем та їх компонентів, який, на відміну від відомих, дає змогу проєктанту інтуїтивно зрозуміло вводити не тільки вхідні дані про структуру, а й про саму архітектуру ПЗ з погляду його надійності, а також автоматизує всі етапи розрахунку показників надійності від етапу побудо-

ви блок-схеми надійності до етапу знаходження розподілу ймовірностей перебування програмної системи у *i*-му стані $P_i(t)$ протягом часу *t* для усіх можливих станів програмної системи.

Практична значущість результатів дослідження – розроблений метод можна використовувати для: автоматизованого формування: умови працездатності, графа станів / переходів, системи диференціальних рівнянь Колмогорова – Чепмена; автоматизованого опрацювання результатів надійнісного проєктування програмних систем довільного рівня складності. Використання розробленого методу дає можливість зменшити вплив людського фактора та імовірність внесення помилки під час розрахунку показників надійності програмних систем на всіх етапах надійнісного проєктування та зменшити час його виконання.

Висновки / Conclusions

Проаналізовано відомі засоби розрахунку показників надійності програмних систем і за результатами встановлено, що для визначення показників надійності програмних систем доцільно використовувати структурно-логічний аналіз блок-схем надійності, оскільки він наочно та найадекватніше відображає розрахунок показників надійності програмної системи загалом та її компонентів зокрема. Визначено також, що процес надійнісного проєктування складних програмних систем загалом, та їх компонентів зокрема, потребує автоматизації усіх його етапів, починаючи від складання RBD і закінчуючи візуалізацією отриманих результатів.

Розроблено метод автоматизації процесу розрахунку показників надійності програмних систем та їх компонентів, який, на відміну від наявних, дає проєктанту змогу інтуїтивно зрозуміло вводити не тільки вхідні дані про структуру, а й про саму архітектуру ПЗ з погляду його надійності, а також автоматизує всі етапи розрахунку показників надійності – від етапу побудови блок-схеми надійності до етапу знаходження розподілу ймовірностей перебування програмної системи у всіх можливих станах.

Запропонований метод дає можливість використовувати в різних комбінаціях методи, алгоритми та програмні засоби, застосовувані для надійнісного проєктування саме програмних систем, І вибрати з них найадекватніші потребам користувача в конкретній ситуації.

Використання розробленого методу дає змогу зменшити вплив людського фактора та імовірності внесення помилки під час розрахунку показників надійності програмних систем на всіх етапах надійнісного проєктування та зменшити тривалість його виконання від 21 % до 146 %.

References

1. ISO/IEC 25010:2011 "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models", 2011. [Online]. Available: https://www.iso.org/obp/ui/#iso: std: iso-iec:25010: ed-1: v1: en
2. Бобало, Ю. Я. та ін. (2013). Математичні моделі та методи аналізу надійності радіоелектронних, електротехнічних та програмних систем: монографія / Ю. Я. Бобало, Б. Ю. Волочій, О. Ю. Лозинський, Б. А. Мандзій, Л. Д. Озірковський, Д. В. Федасюк, С. В. Щербовських, В. С. Яковина. Львів: Вид-во Львівської політехніки, 2013, 300 с.
3. Bharathi, R, & Selvarani, R. (2019). "Software Reliability Assessment of Safety Critical System Using Computational Intelligence". *International Journal of Software Science and Computational Intelligence, 11*(3): 1–25. https://doi.org/10.4018/ijssci.2019070101
4. Teslyuk, V, Sydor, A, Karovič, V ml., Pavliuk, O, & Kazymyra, I. (2021). Modelling Reliability Characteristics of Technical Equipment of Local Area Computer Networks. *Electronics, 10*(8):955. https://doi.org/10.3390/electronics10080955
5. Li, Y F., Huang, H Z., Mi, J. et al. (2022). Reliability analysis of multi-state systems with common cause failures based on Bayesian network and fuzzy probability. *Ann Oper Res, 311*, 195–209. https://doi.org/10.1007/s10479-019-03247-6
6. Ozirkovskyy, L., Volochiy, B., Zmysnyi, M., & Shkiliuk, O. (2020). Synthesis of safe behavior algorithms of radioelectronic systems for critical applications. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 10(1), 28-31. https://doi.org/10.35784/iapgos.913
7. Sydor, A. R., Teslyuk, V. M., & Denysyuk, P. Y. (2014). Recurrent expressions for reliability indicators of compound electropower systems. *Technical Electrodynamics*, *4*, 47-49.
8. Teslyuk, V., Pavliuk, O. & Kazymyra, I. (2021). Model for Determining the Optimal Structure of Hierarchical Asymmetric Branched Computer Networks, *2021 IEEE 32nd International Conference on Microelectronics (MIEL)*, Nis, Serbia, 2021, pp. 259–262, https://doi.org/10.1109/MIEL52794.2021.9569115
9. Karnouskos, S., Sinha, R., Leitão, P., Ribeiro, L., & Strasser, T. I. (2018). "The Applicability of ISO/IEC 25023 Measures to the Integration of Agents and Automation Systems", *IECON 2018 – 44th Annual Conference of the IEEE Industrial Electronics Society*, Washington, DC, USA, 2927–2934. https://doi.org/10.1109/IECON.2018.8592777
10. Catelani, M., Ciani, L., & Venzi, M. (2015). Reliability assessment for complex systems: A new approach based on RBD models, *2015 IEEE International Symposium on Systems Engineering (ISSE)*, Rome, Italy, 286–290. https://doi.org/10.1109/SysEng.2015.7302771
11. Sahinoglu, M., & Ramamoorthy, C. V. (2005). RBD tools using compression, decompression, hybrid techniques to code, decode, and compute reliability in simple and complex embedded systems, Oct. 2005 *IEEE Transactions on Instrumentation and Measurement*, *54*(5), 1789–1799. https://doi.org/10.1109/TIM.2005.855103
12. Bennetts, R. G. (1982). Analysis of Reliability Block Diagrams by Boolean Techniques, June 1982 *IEEE Transactions on Reliability*, *R-31*(2), 159–166. https://doi.org/10.1109/TR.1982.5221283
13. Babu, Naveen, Himagiri, Krishna, V. Vamshi , Kumar, A. Anil , & Ravi, M. (2019). Software Defect Prediction Analysis by Using Machine Learning Algorithms. *International Journal of Recent Technology and Engineering 8*(2S11). 3544-3546. https://doi.org/10.35940/ijrte.b1438.0982s1119
14. Libo, Li, Lessmann, Stefan, & Baesens, Bart (2019). Evaluating Software Defect Prediction Performance: An Updated Benchmarking Study. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3312070
15. Revoori Veeharika Reddy, Nagella Kedharnath, Mandi Akif Hussain, & S. Vidya (2021). Software Defect Estimation using Machine Learning Algorithms. *International Journal of Recent Technology and Engineering (IJRTE), 10*(1), 204–208. https://doi.org/10.35940/ijrte.A5898.0510121
16. Balogun, Abdullateef O., Basri, Shuib, Mahamad, Saipunidzam, Abdulkadir, Said J., & Bajeh, Amos O. (2020). "Impact of Feature Selection Methods on the Predictive Performance of Software Defect Prediction Models: An Extensive Empirical Study". *Symmetry, 12*(7), 1147–1156. https://doi.org/10.3390/sym12071147
17. Lei, T., Xue, J., Wang, Y., Niu, Z., Shi, Z., & Zhang, Y. (2022). WCM-WTrA: A Cross-Project Defect Prediction Method Based on Feature Selection and Distance-Weight Transfer Learning.

Chinese Journal of Electronics, 31, 354–366. https://doi.org/10.1049/cje.2021.00.119
18. Jiarpakdee, J., Tantithamthavorn, C., & Treude, C. (2020). The impact of automated feature selection techniques on the interpretation of defect models. *Empir Software Eng.,* 25, 3590–3638. https://doi.org/10.1007/s10664-020-09848-1
19. Rahul, Hans, & Kaur, Harjot (2020). Opposition-Based Enhanced Grey Wolf Optimization Algorithm for Feature Selection in Breast Density Classification. *International Journal of Machine Learning and Computing, 10*(3), 458–64. https://doi.org/10.18178/ijmlc.2020.10.3.957
20. P. Sridhar & Mehta S. (2021). Stacking Based Ensemble Learning for Improved Software Defect Prediction. *Proceeding of Fifth International Conference on Microelectronics, Computing and Communication Systems,* 167-178. https://doi.org/10.1007/978-981-16-0275-7_14
21. Wei, H., Shan, C., Hu, C., Zhang, Y. and Yu, X. (2019). Software Defect Prediction via Deep Belief Network. *Chinese J. Electron., 28,* 925–932. https://doi.org/10.1049/cje.2019.06.012
22. T. Asano *et al*. (2021). Using Bandit Algorithms for Project Selection in Cross-Project Defect Prediction, *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Luxembourg, 649–653. https://doi.org/10.1109/ICSME52107.2021.00074
23. Tang, S., Huang, S., Zheng, C. Liu, E. Zong, C., & Ding, Y. (2022). A novel cross-project software defect prediction algorithm based on transfer learning, *Tsinghua Science and Technology*, *27*(1), 41–57. https://doi.org/10.26599/tst.2020.9010040
24. Kumar, Lov, Kumar, Mukesh, & Murthy, Lalita Bhanu (2021). An Empirical Study on Application of Word Embedding Techniques for Prediction of Software Defect Severity Level. *Proceedings of 16th Conference on Computer Science and Intelligence Systems (FedCSIS),* 477–484. https://doi.org/10.15439/2021F100
25. Qiao, Lei, Xuesong, Li, Qasim, Umer, & Ping, Guo (2020). Deep Learning Based Software Defect Prediction. *Neurocomputing, 385,* 100–110. https://doi.org/10.1016/j.neucom.2019.11.067
26. Saleh, Albahli (2019). A Deep Ensemble Learning Method for Effort-Aware Just-In-Time Defect Prediction. *Future Internet, 11*(12),246. https://doi.org/10.3390/fi11120246
27. Shi, Ke, Lu, Yang, Chang, Jingfei, & Wei, Zhen (2020). PathPair2Vec: An AST Path Pair-Based Code Representation Method for Defect Prediction. *Journal of Computer Languages, 59,* 100979. https://doi.org/10.1016/j.cola.2020.100979
28. Albahli, Saleh. (2019). A Deep Ensemble Learning Method for Effort-Aware Just-In-Time Defect Prediction. *Future Internet, 11*(12), 246. https://doi.org/10.3390/fi11120246
29. Yakovyna, V. S., Seniv, M. M., Symets, I. I., & Sambir, N. B. (2020). Algorithms and software suite for reliability assessment of complex technical systems. *Radio Electronics, Computer Science, Control, (4), 1*63-177. https://doi.org/10.15588/1607-3274-2020-4-16
30. Seniv, M., Mykuliak, A, & Senechko, A. (2016). Recursive algorithm of traversing reliability block diagram for creation reliability and refuse logical expressions, *2016 XII International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, Lviv, Ukraine, 199–201. https://doi.org/10.1109/MEMSTECH.2016.7507542
31. Bobalo, Y., Seniv, M., Yakovyna, V., & Symets, I. (2018). Method of Reliability Block Diagram Visualization and Automated Construction of Technical System Operability Condition. In: Shakhovska, N., Medykovskyy, M. (eds) *Advances in Intelligent Systems and Computing III. CSIT. Advances in Intelligent Systems and Computing, 871. Springer, Cham.* 599–610. https://doi.org/10.1007/978-3-030-01069-0_43
32. Bobalo, Y., Yakovyna, V., Seniv, M. & Symets, I. (2018). Technique of Automated Construction of States and Transitions Graph for the Analysis of Technical Systems Reliability, *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 314–317. https://doi.org/10.1109/STC-CSIT.2018.8526698
33. Мандзій, Б., Сенів, М., & Гайда, П. (2013). Програмна реалізація моделі надійності технічної резервованої системи з перемикачем. *Комп'ютерні технології друкарства: зб. наук. пр*. Укр. акад. друкарства, 30, 88-96.
34. Bobalo, Y., Yakovyna, V., Seniv, M., & Symets, I. (2019). Techniques of Automated Processing of Kolmogorov – Chapman Differential Equation System for Reliability Analysis of Technical Systems, *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, Polyana, Ukraine, 1–5, https://doi.org/10.1109/CADSM.2019.8779271
35. Yakovyna, V. S., Seniv, M. M., Lytvyn, V. V., & Symets, I. I. (2019). Програмний модуль розв'язування систем диференціальних рівнянь Колмогорова-Чепмена для автоматизації надійнісного проектування. Науковий вісник НЛТУ України, 29(5), 141–146. https://doi.org/10.15421/40290528

M. M. Seniv

Lviv Polytechnic National University, Lviv, Ukraine

A METHOD OF AUTOMATING THE PROCESS OF CALCULATING RELIABILITY INDICATORS OF SOFTWARE SYSTEMS AND THEIR COMPONENTS

The existing means of calculating reliability indicators of software systems are analyzed. It has been established that to determine the reliability indicators of software systems, it is advisable to use the structural-logical analysis of reliability block diagrams, since it clearly and most adequately reflects the process of calculating the reliability indicators of the software system as a whole and its components in particular. Despite the external simplicity of such an analysis, conducting it is not a trivial task, because even building the condition of technical system operability is a difficult task, especially in the case of the presence of many elements with various connections between them, when solving which manually there is a very high probability human error. Also, the construction and visualization of the graph of states / transitions is a non-trivial task, since the number of possible states of the software system depending on the number of elements grows exponentially, and, in turn, increases the complexity of the system of differential equations, the solution of which makes it possible to calculate the necessary reliability indicators. It was determined that the process of reliability design of complex software systems in general, and their components in particular, requires automation of all its stages, starting from the compilation of the reliability block diagram (RBD), and ending with the visualization of the obtained results. A method of automating the process of calculating the reliability indicators of software systems and their components has been developed, which consists of eight steps and, unlike the existing ones, allows the designer to intuitively enter not only input data

about the structure, but also the software architecture itself from the point of view of its reliability, and also automates all stages of calculating reliability indicators, from the stage of constructing a reliability block diagram to the stage of finding the distribution of probabilities of the software system being in all possible states. The proposed method makes it possible to use in various combinations the methods, algorithms and software tools used for the reliability design of software systems and to choose from them the most adequate to the needs of the user in a specific situation. The use of the developed method makes it possible to reduce the influence of the human factor and the probability of making an error in the process of calculating reliability indicators of software systems at all stages of reliability design and to reduce its time by at least 21 %.

Keywords: software, RBD, reliability, states and transitions graph.

Інформація про авторів:

Сенів Максим Михайлович, канд. техн. наук, доцент, кафедра програмного забезпечення. **Email:** maksym.m.seniv@lpnu.ua; https://orcid.org/0000-0003-1044-4628

Цитування за ДСТУ: Сенів М. М. Метод автоматизації процесу розрахунку показників надійності програмних систем та їх компонентів. *Український журнал інформаційних технологій.* 2024, т. 6, № 1. С. 01-08.

Citation APA: Seniv M. M. (2024). A method of automating the process of calculating reliability indicators of software systems and their components. *Ukrainian Journal of Information Tecnology, 6*(1), 01-08. https://doi.org/10.23939/ujit2024.01.01.001