14. Лесная, Ю., Гончаров, Н., & Малахов, С. (2023). Способы развертки параметров серий опорных блоков изображений, как элемент составного ключа экстрактора данных стегоалгоритма. Grail of Science, (23), 254–258. DOI: 10.36074/grail-of-science.23.12.2022.37.

15. Гончаров, М., & Малахов, С. (2025). Моделювання і оцінка обчислювальної складності основних процедур, початкових етапів гібридного стеганоалгоритму. Комп'ютерні науки та кібербезпека, (1), 41-59. DOI: 10.26565/2519-2310-2025-1-04.

16. Гончаров, М., Нарєжній, О., & Малахов, С. (2025). Аналіз передумов для забезпечення консенсусу ресурсів при виконанні процедур вставлення стеганографічних даних. Сучасний захист інформації, 63(3), 37–47. DOI: 10.31673/2409-7292.2025.030518.

17. Honcharov, M., & Malakhov, S. (2025). Assessment of the complexity of input data preprocessing procedures for implementing steganographic transformations. Scientific Trends and Trends in The Context of Globalization. Proceedings of the 9th International Scientific and Practical Conference. Umea, Sweden. 2025. Pp. 275-283. URL: https://archive.interconf.center/index.php/2709-4685/issue/view/19-20.06.2025/262.

Надійшла 26.09.2025

УДК 004.05:004.942
DOI: 10.31673/2409-7292.2025.041206

Добришин Ю.Є.

МЕТОДИКА ОПИСУ ВЗАЄМОЗВ'ЯЗКІВ МІЖ МНОЖИНАМИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ДЕФЕКТІВ, СПОСОБІВ ЇХ ВИЯВЛЕННЯ У ПРОЦЕСІ ДІАГНОСТУВАННЯ ПОШКОДЖЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті розглянуто актуальну науково-прикладну проблему діагностування пошкодженого програмного забезпечення, що зазнало впливу кібератак, у контексті підвищення кіберстійкості сучасних інформаційних систем. Метою дослідження є формалізація процесу діагностування пошкодженого програмного забезпечення внаслідок дії кібератак, що забезпечує системне представлення взаємозв'язків між множинами програмних модулів, дефектів, способів їх виявлення та методів відновлення. Для досягнення поставленої мети застосовано методи системного аналізу, математичного моделювання, теорії множин і формальної логіки, які дозволили описати технологічний процес діагностування у вигляді сукупності взаємопов'язаних формальних відношень. Розроблена формалізована методика дає можливість перейти до автоматизованого діагностування пошкодженого програмного забезпечення. Вона забезпечує систематизацію відомостей про дефекти, підвищує точність локалізації порушень, знижує трудомісткість аналізу та скорочує час відновлення після інцидентів. Математичний апарат, що запропонований у роботі, дозволяє відображати не лише факт появи дефекту, але й його властивості, взаємозв'язки з іншими дефектами, умови виникнення та можливі способи усунення. Результати дослідження мають теоретичне і практичне значення. Запропонована методика сприяє підвищенню ефективності відновлення працездатності програмного забезпечення після кібератак, мінімізації фінансових і репутаційних втрат, а також покращенню якості та надійності функціонування інформаційних систем. Вона створює передумови для подальших досліджень у напрямі розробки інтелектуальних діагностичних систем, заснованих на машинному навчанні та логічному моделюванні взаємозв'язків між дефектами.

Ключові слова: автоматизація діагностики, відновлення працездатності систем, діагностування програмного забезпечення, кібербезпека, кібератаки, математичне моделювання, формалізована модель.

Вступ і формулювання проблеми

На теперішній час значно зростає кількість та складність кібератак, які становлять критичну загрозу для функціонування інформаційних систем. Еволюція технологій кіберзлочинності, зокрема застосування штучного інтелекту для генерації поліморфного шкідливого коду, здатного створювати тисячі варіантів і уникати виявлення традиційними засобами захисту, вимагає переосмислення існуючих підходів до забезпечення кібербезпеки.

У цьому контексті особливого значення набуває діагностування пошкодженого програмного забезпечення, яке дозволяє оцінити стан компонентів системи після впливу кібератак і визначити оптимальні шляхи їх відновлення. Критичним аспектом діагностики є систематизація виявлених пошкоджень через класифікацію та кодування дефектів програмного забезпечення, що дозволяє стандартизувати процес документування інцидентів та забезпечити ефективний обмін інформацією між фахівцями. Вона є необхідною

передумовою для автоматизації процесів діагностики та створення інтелектуальних систем підтримки прийнятих рішень, які можуть ефективно аналізувати та категоризувати різноманітні типи можливостей програмного забезпечення.

Сама ж діагностика пошкодженого програмного забезпечення після кібератак являє собою комплексний процес, що виходить за межі простого виявлення шкідливого коду і включає детальний аналіз компрометованих компонентів, визначення масштабів пошкодження, реконструкцію ланцюга подій атаки та забезпечення юридичного значення зібраних доказів. Ефективність діагностики безпосередньо впливає на можливість своєчасного відновлення функціональності систем, мінімізацію фінансових та репутаційних втрат, а також попередження подібних інцидентів у майбутньому, що обумовлює критичну важливість розробки та вдосконалення методів діагностики пошкодженого програмного забезпечення.

Технологічний процес діагностування належить до складних процесів, що відбуваються у багаторівневих автоматизованих системах, і ставить за мету виявлення пошкоджень та несправностей програмного забезпечення на момент їх виявлення. При цьому процес є трудомістким і складним для автоматизації через відсутність адекватних математичних моделей.

Це обумовлює необхідність застосування формалізованих методів діагностики, які базуються на математичних і логічних моделях. Такі моделі дозволяють системно описати властивості дефектів, взаємозв'язки між ними та відповідні способи їх усунення. Окрім того формалізація діагностування пошкодженого програмного забезпечення може класифікувати дефекти за їх типами, визначати причину їх появи, а також прогнозувати можливі наслідки виходу з ладу програмного забезпечення з подальшим вибором оптимального способу відновлення.

Актуальність дослідження формалізованих моделей діагностування посилюється у зв'язку з тим, що кібератаки стають дедалі складнішими, а традиційні методи відновлення програмного забезпечення не завжди забезпечують швидке повернення системи у працездатний стан. Розвиток інтелектуальних методів аналізу, таких як зокрема машинного навчання, методів прогнозування дефектів та евристичних алгоритмів надає нові можливості для автоматизації процесу діагностування й оптимізації процедур відновлення.

Отже, наукова проблема полягає у відсутності формалізованого підходу щодо діагностування пошкодженого програмного забезпечення після впливу кібератак, який би забезпечував об'єктивне виявлення, класифікацію та прогнозування дефектів з метою вибору оптимальних способів відновлення системи.

Аналіз літератури

Питання опису та розробки формальної моделі дефектів, яка використовує математичні або логічні структури (наприклад, множини, функції, графи, онтології, логічні відношення) для опису: типів дефектів їх властивостей, зв'язків між дефектами, способів виявлення дефектів та засобів їх усунення присвячено значна кількість наукових праць закордонних та вітчизняних фахівців. Метою таких робіт є спроба формалізувати процес діагностики, підвищити точність, автоматизувати аналіз появи дефектів та надати пропозиції щодо прийняття необхідних рішень щодо способів їх усунення.

Так в окремих наукових працях [1-4] надається огляд та аналіз методів та моделей оцінки якості програмного забезпечення в інформаційно-комунікаційних системах та описується таксономія помилок, які виникають на етапі проєктування програмного забезпечення.

У роботах надається класифікація помилок, що виникають у наслідку людського фактору та розробки помилкових архітектурних рішень інформаційних систем, логіки бізнесу тощо. Автори робіт стверджують, що такий підхід дуже корисний, якщо виконується формалізація помилок програмного забезпечення на рівні дизайн-дефектів. Так, наприклад, у науковій праці [2] розглядається графова модель щодо опису дефектів у складних технічних системах, описуються причинно-наслідкові зв'язки для аналізу причин виникнення дефектів.

© Добришин Ю.Є. (2025) Методика опису взаємозв'язків між множинами програмного забезпечення, дефектів, способів їх виявлення у процесі діагностування пошкодженого програмного забезпечення. Сучасний захист інформації, 4(64), 59 – 67.
https://doi.org/10.31673/2409-7292.2025.041206

Окремі наукові фахівці на підставі досліджень процесів діагностики працездатності програмного забезпечення представляють концептуальну модель життєвого циклу програмного забезпечення, що орієнтоване на безпеку. У науковій праці [5], зазначена модель включає етапи виявлення дефектів, оцінки ризиків, можливості формального відображення етапів та властивостей, хоча не всі відношення між дефектами, а саме, домінування, еквівалентність сформульовані математично.

Інтерес представляють окремі наукові праці, де аналізуються відмови у роботі програмного забезпечення. Так у науковій праці [6] описується підхід до аномалій та відмов роботи програмного забезпечення. У науковій роботі представлена семантика, яка описує помилки роботи програмного забезпечення, дефект, поняття відмови та впливу дефектів програмного забезпечення на працездатність інформаційної системи.

У роботі [7] з метою моделювання сумісності дефектів, способів їх виявлення та відновлення, наведені шаблони дефектів програмного забезпечення, на підставі яких показано як будувати формальні моделі взаємозв'язків між вразливістю програмного забезпечення та їх наслідками.

Питанням прогнозування дефектів програмного забезпечення із застосуванням методів машинного навчання присвячені наукові праці [8,9,10]**.** Автори наукових праць аналізують проблеми застосування методів машинного навчання для оцінювання та прогнозування дефектів програмного забезпечення та на підставі цього роблять практичні висновки щодо складу, точності та стабільності формальних моделей дефектного програмного забезпечення. У науковій праці [11] розглянуто метод базового статичного аналізу шкідливого програмного забезпечення, що базується на аналізі заголовків файлів, що виконуються та динамічних бібліотек, які побудовані з використанням спеціалізованого формату Portable Executable. Це дозволяє виявляти шкідливі компоненти програмного забезпечення та оцінювати їх вплив на його працездатність.

В окремих роботах [12-14]. формалізація процесів діагностування будується на підставі науковій класифікації та системи іменування груп об'єктів за певними принципами, найчастіше ієрархічними. Ця таксономія, розроблена на основі реальних дефектів на підставі аналізу десяті програмних систем, що використовує різні методи для класифікації дефектів. Вона дозволяє систематизувати дефекти та сприяє покращенню процесів тестування та сертифікації програмного забезпечення.

У роботі [12]. представлена таксономія дефектів, що виникають на етапі збору вимог. Вона допомагає виявляти та класифікувати дефекти на ранніх етапах розробки, що сприяє підвищенню якості програмного забезпечення. кінцевого продукту. Інші корисні моделі формалізації та підходи щодо опису дефектів розглядаються у наукових роботах [13 14]. Авторами аналізуються класифікація дефектів коду, за допомогою яких можна виділяти атрибути якості програмного забезпечення, а також атрибути якості програмного забезпечення, включаючи функціональність, зручність використання, надійність, продуктивність. Серед зазначених робіт інтерес представляє робота [14], де наведені моделі, які описують дефекти та методи їх виявлення, що можуть бути використані для автоматизації процесу діагностики та усунення дефектів.

Таким чином аналіз сучасних наукових досліджень свідчить про активний розвиток напрямів, пов'язаних із формалізацією процесів діагностики та усунення дефектів програмного забезпечення. Існуючі роботи охоплюють широке коло підходів щодо розробки формалізованих моделей від побудови графових моделей до застосування різних структур та методів машинного навчання для прогнозування появи дефектів.

Дослідники прагнуть описати типи помилок, їх властивості, взаємозв'язки та вплив на працездатність програмних систем, що дозволяє частково автоматизувати процес виявлення та класифікації дефектів. Особливу увагу приділено дослідження дефектів, аналізу аномалій і відмов у роботі програмного забезпечення, а також побудові концептуальних і якісних

© Добришин Ю.Є. (2025) Методика опису взаємозв'язків між множинами програмного забезпечення, дефектів, способів їх виявлення у процесі діагностування пошкодженого програмного забезпечення. Сучасний захист інформації, 4(64), 59 – 67.
https://doi.org/10.31673/2409-7292.2025.041206

моделей, орієнтованих на безпеку та надійність програмних систем. Разом з тим, узагальнення результатів проведених досліджень показує, що відсутня єдина формальна модель діагностики пошкодженого програмного забезпечення, яка б уніфіковано відображала структуру дефектів, причинно-наслідкові зв'язки між ними, механізми їх виявлення та усунення. Існуючі підходи переважно зосереджуються на окремих аспектах класифікації та прогнозуванні або оцінці ризиків виходу з ладу програмного забезпечення без цілісного формального опису процесу діагностики. Це ускладнює інтеграцію різних методів у єдине діагностичне середовище та знижує точність і відтворюваність результатів аналізу.

Отже, наукова проблема полягає у розробці формалізованої моделі діагностики пошкодженого програмного забезпечення, яка спирається на математичні та логічні засоби (множини, графи, логічні відношення) що дозволяє описати повний життєвий цикл дефекту від його виникнення до усунення. Така модель має забезпечити підвищення точності виявлення дефектів, автоматизацію процесів діагностики, а також покращення якості, безпеки й надійності програмних систем.

Мета та завдання дослідження

Метою статті ставиться завдання розробити формалізовану методику діагностування пошкодженого програмного забезпечення, на підставі застосування математичної моделі, що описує функціональні залежності між об'єктами, які приймають участь у процесі діагностування. Застосування такої моделі дозволяє здійснити класифікацію та аналіз характеристик та взаємозв'язків між технологічними об'єктами, які беруть участь у процесі діагностики пошкодженого програмного забезпечення внаслідок впливу кібератак. Це також дає змогу виявити якісні взаємозв'язки між операціями, що стосуються властивостей дефектів і методів їх усунення, а також формалізувати технологічну послідовність дій, спрямованих на відновлення працездатності пошкодженого програмного забезпечення внаслідок впливу кібератак.

Основна частина

Для формалізованого опису технологічного процесу діагностування пошкодженого програмного забезпечення, необхідно розглянути відношення між множиною програмного забезпечення $\{PR\}$ множиною дефектів $\{GP\}$, множиною способів їх відновлення $\{VP\}$, та способі виявлення $\{VA\}$, які породжують аналітичні взаємозв'язки, що доступні для математичної інтерпретації, це в свою чергу дозволяє перейти до формалізованого опису технології діагностування пошкодженого програмного забезпечення.

Аналіз пошкодженого програмного забезпечення внаслідок дії кібератак показує, що будь-яка програма P_z, де $z \in \{PR\}$, яка перевіряється на наявність дефектів може бути описана кортежом:

$$P_z = \{ P_i, G, CT_p\}, \tag{1}$$

де: P_i загальні відомості щодо пошкодженого програмного забезпечення, внаслідок дії кібератак; G_g, де $g_f \in \{GP\}$, множина дефектів пошкодженого програмного забезпечення; S_p – структура пошкодженого програмного забезпечення.

Загальні відомості щодо пошкодженого програмного P_i де i $\in \{PM\}$ представимо у вигляді множини програмних модулів та можливих їх властивостей, приклад яких наведений у таблиці 1.

$$P_i = \{ S_p^1, S_p^2, S_p^3 \dots\dots\dots, S_p^i\} \tag{2}$$

У свою чергу характеристики кожного з дефектів пошкодженого програмного забезпечення внаслідок впливу кібератак g_f, де f $\in \{GP\}$ можна описати за допомогою їх властивостей (таблиця 2), а саме:

$$g_f = \{ S_g^1, S_g^2, S_g^3 \dots\dots\dots, S_g^f\} \tag{3}$$

Таблиця 1

Приклад основних властивостей пошкодженого програмного забезпечення (фрагмент)

Назва елементу	**Властивість (функціональність)**	**Позначення властивості (функціональності)**
Пошкоджене програмне забезпечення P_z де $z \in \{PR\}$	Системне програмне забезпечення	$[S_{pz}^1]$
	Прикладне програмне забезпечення	$[S_{pz}^2]$
	Інструментальне програмне забезпечення	$[S_{pz}^3]$
Програмний модуль P_i де $i \in \{PM\}$	Модулі ядра	$[S_p^1]$
	Системні модулі	$[S_p^2]$
	Модулі програмних бібліотек	$[S_p^3]$
	Модулі драйверів пристроїв	$[S_p^4]$
	Інтерфейсні модулі	$[S_p^5]$
	Модулі бізнес-логіки	$[S_p^6]$
	Модулі бази даних	$[S_p^7]$
	Модулі доступу до даних	$[S_p^8]$
	Комунікаційні модулі	$[S_p^9]$

Таблиця 2

Приклад основних властивостей дефектів пошкодженого програмного забезпечення (фрагмент)

Назва елементу	**Властивість (функціональність)**	**Позначення властивості (функціональності)**
Дефект g_f, де $f \in \{GP\}$	Втрата контролю над системою	$[S_g^1]$
	Вилучення критичних файлів	$[S_g^2]$
	Зміна критичних файлів	$[S_g^3]$
	Відсутність відповіді системи на запити	$[S_g^4]$
	Втрата доступу до функцій системи	$[S_g^5]$
	Системні збої у роботі програмного забезпечення	$[S_g^6]$
	Системні аварії у роботі програмного забезпечення	$[S_g^7]$
	Невідповідність логіки роботи програми	$[S_g^8]$
	Витік оперативної пам'яті системи	$[S_g^9]$
	Невірне використання процесорного часу	$[S_g^{10}]$
	Порушення роботи з базами даних	$[S_g^{11}]$
	Відсутність патчів безпеки	$[S_g^{12}]$
	Некоректне оновлення програмного забезпечення	$[S_g^{13}]$
	Зміна інформації у системних журналах	$[S_g^{14}]$
	Видалення системних журналів	$[S_g^{15}]$
	Порушення роботи мережевих протоколів	$[S_g^{16}]$
	Порушення конфіденційності	$[S_g^{17}]$
	Пошкоджена цільність даних	$[S_g^{18}]$
	Порушення цільності програмного коду	$[S_g^{19}]$
	Небезпека витоку даних	$[S_g^{20}]$
	Витік даних	$[S_g^{21}]$
	Пошкодження даних	$[S_g^{22}]$
	Шифрування файлів для вимагання	$[S_g^{23}]$

CT_p структуру пошкодженого програмного забезпечення внаслідок дії кібератак представимо графом, у якому вершинами будуть виступати дефекти g_f, де f ∈ $\{GP\}$, а ребра множина взаємозв'язків та відношень між дефектами.

Таким чином можна визначити ряд відношень, які будуть задані певними властивостями, які породжують взаємозв'язки, доступні математичні інтерпретації, що дозволяють формалізувати поняття вибору та порівняння рішень під час проведення діагностування пошкодженого програмного забезпечення.

Серед таких відношень доцільно виділити наступні відношення:

- сумістності;
- еквівалентності;
- домінування;
- причинно-обумовлене відношення між дефектами;
- слідування.

Сумісність «↔» програмного забезпечення P_f, де $f \in \{N\}$ що діагностується та дефекту $g_j, j \in \{G\}$де досягається за умови виконання умови:

$$\forall_N P_f \ \forall_G g_f \ \forall_G g_j \ \exists S_p \ \exists S_g \ \ni P\{\left(P_f\left[S_p^i\right] = g_f\left[S_g^i\right]\right) \wedge \left(P_f\left[S_p^i\right] = g_j\left[S_g^i\right]\right)\} \rightarrow \left(P_f \ \leftrightarrow g_j\right), \quad (4)$$

де: P_i – пошкоджене програмне забезпечення; g_j , g_f – дефекти пошкодженого програмного забезпечення; S_p, S_g – властивості програмного забезпечення та дефекту.

Під відношенням еквівалентності будемо зазначати рівність технологічних властивостей дефектів пошкодженого програмного забезпечення.

Вказане відношення позначимо як (≈).

$$\forall g_{fx} \forall g_{fy} \exists S_{gx} \exists S_{gy} \ni P\{g_{fx}\left[S_{gx}\right] = g_{fy}\left[S_{gy}\right] \wedge (g_{fx} \neq g_{fy}\} \rightarrow g_{fx} \approx g_{fy.} \quad (5)$$

Поява такого відношення вважається при наявності однакових властивостей дефектів пошкодженого програмного забезпечення.

На прикладі властивостей основних дефектів пошкодженого програмного забезпечення, які приведені у таблиці 2, відношення еквівалентності (≈) для дефекту буде записане таким чином:

$$\forall g_f^1 \forall g_f^2 \exists S_g^{18} \ \exists S_g^{22} \ \ni P\{g_f^1\left[S_g^{18}\right] = g_f^2\left[S_g^{22}\right] \wedge (g_f^1 \neq g_f^2\} \rightarrow g_f^1 \approx g_f^2 \quad (6)$$

Визначимо окремий порядок виявлення дефектів під час проведення діагностування пошкодженого програмного забезпечення, шляхом введення поняття відношення домінування.

Під відношенням домінування будемо зазначати, що два дефекта будуть перевірятися у певної послідовності, в разі якщо властивості одного дефекту перевершують властивості іншого дефекту, тобто один дефект g_f^1 домінує на іншим дефектом g_f^2 .

Відношення домінування позначимо як (≫). Враховуючі зазначене, можна записати відношення домінування наступним математичним виразом:

$$\forall g_{fx} \forall g_{fy} \ \exists \ S_{gx} \ \exists \ S_{gy} \ \ni P\{[(P(g_{fx}\left[S_{gx}\right] = 1) \wedge [(P\left(g_{fy}\left[S_{gy}\right] = 0\right))]\} \rightarrow \ \left[g_{fx} \gg g_{fy}\right]. \quad (7)$$

Враховуючи властивості основних дефектів пошкодженого програмного забезпечення, які приведені у таблиці 2, відношення домінування (≫) може бути представлено таким чином:

$$\forall \ g_f^1 \ \forall \ g_f^2 \ \exists S_g^6 \ \exists S_g^7 \ \ni P\{[(P(\ g_f^1\left[S_g^7\right] = 1) \wedge [(P\left(g_f^2\left[S_g^6\right] = 0\right))]\} \rightarrow \left[g_f^1 \gg g_f^2\right]. \quad (8)$$

Наприклад під час діагностування пошкодженого програмного забезпечення був виявлений дефект, який не може бути відновлений без повного перевстановлення та конфігурування програмного забезпечення, що свідчить про не доцільності подальшого діагностування.

© Добришин Ю.Є. (2025) Методика опису взаємозв'язків між множинами програмного забезпечення, дефектів, способів їх виявлення у процесі діагностування пошкодженого програмного забезпечення. Сучасний захист інформації, 4(64), 59 – 67.
https://doi.org/10.31673/2409-7292.2025.041206

Сформулюємо ще одне відношення, яке є одним з важливих у процесі діагностування пошкодженого програмного забезпечення внаслідок впливу кібератак, це відношення обумовлено тим, що поява одного дефекту може визначати існування іншого. Таке явище пов'язане з причинно-обумовленим відношенням між дефектами пошкодженого програмного забезпечення. Вказане відношення позначимо як (∼). Враховуючі зазначене, можна записати вказане відношення наступним математичним виразом:

$$\forall_{pm} P_i \exists g_f^1 \exists g_f^2 \exists S_p \exists S_g \ni P\{(P_i[S_p^1] \leftrightarrow g_f^1[S_g^1]) \wedge (P_i[S_p^2] \leftrightarrow g_f^2[S_g^2]) \wedge (g_f^1 \neq g_f^2)\} \rightarrow (g_f^1 \sim g_f^2). \quad (9)$$

На підставі приведених властивостей дефектів (таблиця 2) причинно-обумовлене відношення між дефектами можна записати наступним чином:

$$\forall_{pm} P_i \exists g_f^1 \exists g_f^2 \exists S_p \exists S_g \ni P\{(P_i[S_p^2] \leftrightarrow g_f^1[S_g^2]) \wedge (P_i[S_p^6] \leftrightarrow g_f^2[S_g^6]) \wedge (g_f^1 \neq g_f^2)\} \rightarrow (g_f^1 \sim g_f^2). \quad (10)$$

Під відношенням слідування будемо розуміти послідовність діагностування дефектів пошкодженого програмного забезпечення, виходячи з властивостей дефекту.

Тобто першим в технологічному процесі діагностування буде вибиратися найбільш важливий дефект, який суттєво пливає на працездатність програмного забезпечення, а далі можуть діагностуватися дефекти при яких пошкоджене програмне забезпечення може використовуватися певний час. Позначимо відношення слідування як (¬).

Наприклад, під час діагностування у першу чергу визначаються причини виникнення системних аварій у роботі програмного забезпечення, а далі порушення цільності його програмного коду.

Відношення слідування можна записати наступним виразом:

$$\forall_{Gp} g_f^1 \ \forall_{Gp} g_f^2 \ \exists S_g \ni P\{((g_f^1[S_g^1] > g_f^2[S_g^2]) \rightarrow (g_f^1 \neg g_f^2)\}. \quad (11)$$

На підставі приведених властивостей дефектів (таблиця 2) технологічна послідовність усунення дефектів визначається із істинності наступних умов:

$$\forall g_f^1 \ \forall g_f^2 \ \exists S_g^1 \ni S_g^2 \ni P\{[P(g_f^1[S_g^1] > g_f^2[S_g^4])\} \rightarrow [g_f^1 \neg g_f^2]. \quad (12)$$

Як в наслідок виникає властивості транзитивності дефектів:

$$([g_f^1 \leftrightarrow g_f^2] \wedge [g_f^2 \leftrightarrow g_f^3]) \rightarrow [g_{f_i}^1 \leftrightarrow g_f^3]. \quad (13)$$

Дане відношення дозволяє розмістити всі дефекти в порядку убування, з урахування рані виявлених відношень та властивостей кожного дефекту. Очевидно на першому місті будуть знаходитися те дефекти, властивості яких найбільш суттєво впливають на працездатність програмного забезпечення.

Інші дефекти, які можна вважати неосновними і наявність яких тимчасово дозволяється, можуть розглядатися як умовно допустимі і пов'язані з основними дефектами певними закономірностями. Тобто має місце залежність:

$$g_{f\text{н}i} = F(g_{foi}), \quad (14)$$

де, $g_{f\text{н}i}$ величина неосновного дефекту, g_{foi} величина основного дефекту.

Тобто втрати від величини неосновного дефекту невеликі, тому в окремих випадках їх можна не враховувати. Вказана умова дозволяє зменшити число дефектів, які потребують перевірки, що суттєво зменшує час на проведення діагностування.

Під час проведення діагностування для пошкодженого програмного забезпечення, серед багатьох способів відновлення (табл. 3), необхідно визначити підмножину способів C_j де $j \in \{VP\}$, які за своїми характеристиками спроможні відновити програмний модуль.

© Добришин Ю.Є. (2025) Методика опису взаємозв'язків між множинами програмного забезпечення, дефектів, способів їх виявлення у процесі діагностування пошкодженого програмного забезпечення. Сучасний захист інформації, 4(64), 59 – 67.
https://doi.org/10.31673/2409-7292.2025.041206

Таблиця 3

Приклад основних властивостей способів відновлення пошкодженого програмного забезпечення (фрагмент)

Назва елементу	**Властивість (функціональність)**	**Позначення властивості (функціональності)**
Спосіб відновлення C_j де $j \in \{VP\}$	Повне переустановлення операційної системи	$[S_v^1]$
	Повне переустановлення прикладного програмного забезпечення	$[S_v^2]$
	Повне переустановлення інструментального програмного забезпечення	$[S_v^3]$
	Заміна всіх облікових даних	$[S_v^4]$
	Проведення аудиту системи	$[S_v^5]$
	Встановлення оновлень безпеки	$[S_v^6]$
	Відновлення програмного забезпечення з резервної копії	$[S_v^7]$
	Усунення пошкоджених програмних модулів	$[S_v^8]$
	Оновлення програмного забезпечення	$[S_v^9]$
	Сканування антивірусним програмним забезпеченням	$[S_v^{10}]$
	Видалення шкідливого програмного забезпечення	$[S_v^{11}]$
	Аудит мережевої активності	$[S_v^{12}]$
	Використання спеціальних утиліт для дешифрування	$[S_v^{13}]$
	Повне очищення системи	$[S_v^{14}]$
	Аудит конфігурацій програмних модулів	$[S_v^{15}]$
	Виявлення змін у коді або налаштуваннях програмних модулів	$[S_v^{16}]$
	Повернення роботи програмного забезпечення до контрольної точки	$[S_v^{17}]$
	Зміна параметрів безпеки	$[S_v^{18}]$
	Проведення тестування програмних модулів на проникнення	$[S_v^{19}]$
	Перевірка на наявність шкідливих компонентів	$[S_v^{20}]$
	Верифікація цілісності даних	$[S_v^{21}]$

Зазначимо умови, які дозволяють визначати відношення сумісності (↔) між елементами технологічного процесу. Так, наприклад відношення сумісності між дефектним програмним модулем P_i та способом відновлення формально можна описати наступним чином:

$$\forall P_i \forall C_j \exists S_p \exists S_v \ni P\{(P_i\,[S_p^7] = C_j[S_v^7]) \wedge (P_i\,[S_p^7] = C_j[S_v^{17}] \wedge (P_i\,[S_p^7] = C_j[S_v^{21}])\} \rightarrow [P_i \leftrightarrow C_j], \quad (15)$$

де: P – предикат, що характеризує істинність відношення.

Сформулюємо умови сумісності між дефектом g_f, де f ∈ $\{GP\}$ та способом його відновлення. У формальному вигляді це можливо записати таким чином:

$$\forall g_f\ \forall C_j\ \exists S_g\ \exists S_v \ni P\ \{(g_f\ [S_g^{11}] = C_j[S_v^7])\} \rightarrow [g_f \leftrightarrow C_j]. \quad (16)$$

Виходячи з цього не складно визначити, що на підставі приведених умов можливо побудувати властивість транзитивності між пошкодженим програмним забезпеченням, способом його відновлення та дефектом:

$$P\ ([P_i \leftrightarrow C_j] \wedge [g_f \leftrightarrow C_j]) \rightarrow [P_i \leftrightarrow g_f]. \quad (17)$$

Під час проведення діагностування дефектів пошкодженого програмного забезпечення однієї з важливих ознак класифікації є розподіл дефектів у залежності від способу їх виявлення $\{S_{va}\}$. Такі ознаки забезпечують оптимальне використання спеціалізованого програмного та технічного забезпечення, сприяють оптимальному застосуванню операцій щодо налаштування системи моніторингу та виявлення кібератак (табл.4).

© Добришин Ю.Є. (2025) Методика опису взаємозв'язків між множинами програмного забезпечення, дефектів, способів їх виявлення у процесі діагностування пошкодженого програмного забезпечення. Сучасний захист інформації, 4(64), 59 – 67.
https://doi.org/10.31673/2409-7292.2025.041206

Таблиця 4

Приклад основних властивостей способів виявлення дефектів пошкодженого програмного забезпечення (фрагмент)

Назва елементу	**Властивість (функціональність)**	**Позначення властивості (функціональності)**
Спосіб виявлення S_{va_i} де $i \in \{VA\}$	Аналіз журналів безпеки	$[S_{va}^1]$
	Перевірка цілісності файлів	$[S_{va}^2]$
	Аналіз трафіку мережі	$[S_{va}^3]$
	Аналіз вразливостей програмного забезпечення	$[S_{va}^4]$
	Тестування на проникнення	$[S_{va}^5]$
	Аналіз наявності шкідливих програм	$[S_{va}^6]$
	Моніторинг поведінки користувачів	$[S_{va}^7]$

У формальному вигляді умови сумісності між дефектом g_f, де f ∈ $\{GP\}$ та способом його виявлення S_{va_i} де і ∈ $\{VA\}$, можна записати таким чином:

$$\forall g_f \forall S_{vai} \exists S_g \exists S_{va} \ni P\{(g_f\ [S_g^i] = S_{vai}[S_{va}^f])\} \to [g_f\ \leftrightarrow\ S_{vai}]. \quad (18)$$

Виходячи з даних приведених у таблицях 2 та 4 відношення сумісності між дефектом та способом його відновлення формально можна описати наступним чином:

$$\forall g_f \forall S_{vai} \exists S_g \exists S_{va} \ni P\{(g_f\ [S_g^{18}] = S_{vai}[S_{va}^2])\} \to [g_f\ \leftrightarrow\ S_{vai}]. \quad (19)$$

Відношення, які були визначені між множиною програмного забезпечення $\{PR\}$, множиною дефектів $\{GP\}$, множиною способів їх відновлення $\{VP\}$, та способів їх виявлення породжують аналітичні взаємозв'язки, доступні для математичної інтерпретації, що дозволяє перейти до автоматизованої побудови технологічного процесу діагностування пошкодженого програмного забезпечення.

Висновки і рекомендації

Проведене дослідження підтверджує, що формалізація процесу діагностування пошкодженого програмного забезпечення після впливу кібератак є необхідною умовою для підвищення ефективності та достовірності відновлення його працездатності. Запропонований підхід базується на використанні математичної моделі, яка описує взаємозв'язки між множинами програмного забезпечення, дефектів, способів їх виявлення та методів відновлення. Це дозволяє не лише структурувати процес діагностики, але й визначити пріоритетність усунення дефектів відповідно до їхнього впливу на функціональність системи.

Розроблені формальні відношення – сумісності, еквівалентності, домінування, причинно-обумовленості та слідування – створюють основу для побудови аналітичних залежностей, доступних для математичної інтерпретації.

Застосування таких залежностей уможливлює автоматизацію процесу діагностування, скорочення часу виявлення дефектів та мінімізацію ризиків помилкових рішень. Таким чином, формалізована методика діагностування пошкодженого програмного забезпечення забезпечує системність підходу до відновлення програм після кібератак, підвищує надійність інформаційних систем критичної інфраструктури та сприяє створенню уніфікованих технологічних процедур їх відновлення.

Перелік посилань

1. Hamretskyi, R. M., & Hnatiuk, V. O. (2024). Огляд та аналіз методів та моделей оцінки якості програмного забезпечення в інформаційно-комунікаційних системах. Інформаційні технології, кібербезпека, 64(4), 445–447. https://doi.org/10.18372/2310-5461.63.19752.

2. Meng, X., Jing, B., Wang, S., Pan, J., Huang, Y., & Jiao, X. (2023). Fault knowledge graph construction and platform development for aircraft PHM. Sensors (Basel), 24(1), 231. https://doi.org/10.3390/s24010231.

© Добришин Ю.Є. (2025) Методика опису взаємозв'язків між множинами програмного забезпечення, дефектів, способів їх виявлення у процесі діагностування пошкодженого програмного забезпечення. Сучасний захист інформації, 4(64), 59 – 67.
https://doi.org/10.31673/2409-7292.2025.041206

3. Agrawal, T., Walia, G. S., & Anu, V. K. (2024). Development of a software design error taxonomy: A systematic literature review. SN Computer Science, 5, 467. https://doi.org/10.1007/s42979-024-02797-2.

4. Verma, A., Agarwal, A., Rathore, M., Bisht, S., & Singh, D. (2023). Preferential selection of software quality models based on a multi-criteria decision-making approach. International Journal of Software Innovation (IJSI), 11(1), 1–13. https://doi.org/10.4018/IJSI.315739.

5. Kordunova, Yu. S. (2023). Analysis and development of a conceptual model for lifecycle management of specialized safety-oriented software. Ukrainian Journal of Information Technology, 5(2), 72–78. https://doi.org/ 10.23939/ujit2023.02.072.

6. Duarte, B. B., Falbo, R. A., Guizzardi, G., Guizzardi, R., & Souza, V. E. S. (2021). An ontological analysis of software system anomalies and their associated risks. Data & Knowledge Engineering, 134, 101892. https://doi.org /10.1016/j.datak.2021.101892.

7. Hu, X., Chen, J. M., & Li, H. (2024). Software security vulnerability patterns based on ontology. Journal of Beijing University of Aeronautics and Astronautics, 50(10), 3084–3099. https://doi.org/10.13700/j.bh.1001-5965. 2022.0783.

8. Yakovyna, V. S., & Symets, I. I. (2021). Прогнозування дефектів програмного забезпечення ансамблем нейронних мереж. Науковий вісник НЛТУ України, 31(6), 104–111.

9. Biletskyi, T. P., & Fedasiuk, D. V. (2021). Прогнозування дефектів у програмному забезпеченні алгоритмами глибинного навчання CNN та RNN. Науковий вісник НЛТУ України, 31(2), 114–120.

10. Khil, O. S., & Yakovyna, V. S. (2023). Аналіз проблеми застосування методів машинного навчання для оцінювання та прогнозування дефектів програмного забезпечення. Науковий вісник НЛТУ України, 33(3), 110–116.

11. Yehorov, S. V., & Shkvarnytska, T. Yu. (2020). Розширений метод аналізу шкідливого програмного забезпечення з метою створення сигнатур. Вісник Університету «Україна», 1(24), 161–169.

12. Alshazly, A. A., Elfatatry, A. M., & Abougaba, M. S. (2024). Detecting defects in software requirements specification. Alexandria Engineering Journal, 53(3), 513–527.

13. Rumbutis, L., Slotkene, A., & Pliuskuviene, B. (2024). Визначення атрибутів якості програмного забезпечення на основі прогнозування дефектів коду: систематичний огляд літератури. Нові тенденції в комп'ютерних науках, 2(1), 57–68. https://doi.org/10.3846/ntcs.2024.21305.

14. Oivo, M., Abrahamsson, P., Corral, L., & Russo, B. (Eds.). (2020). Product-Focused Software Process Improvement – 16th International Conference, PROFES 2015: Proceedings (pp. 380–396). Springer Verlag. https://doi.org/10.1007/978-3-319-26844-6_28.

Надійшла 28.09.2025

УДК 004.738.1:621.39
DOI: 10.31673/2409-7292.2025.041207

Дрововозов В.І.,Аль-Шаммарі А.А.А.

МОДЕЛЬ ВИЗНАЧЕННЯ МАРШРУТУ ЯКОСТІ ОБСЛУГОВУВАННЯ (QOS) ТА ЙОГО ЗАТРИМОК ПРИ ВИКОРИСТАННІ МОДЕЛІ БЕЗДРОТОВОЇ МЕРЕЖІ

У статті запропоновано нову математичну модель маршрутизації для бездротових мереж, спрямовану на забезпечення якості обслуговування (QoS) за умов динамічного, самоподібного трафіку. Модель ґрунтується на міжрівневому підході, що інтегрує фізичний, канальний та мережевий рівні стеку протоколів, і використовує релаксацію Лагранжа для декомпозиції складної задачі оптимізації з метою мінімізації кінцевих затримок, втрат пакетів та енергоспоживання. Для пошуку квазіоптимальних рішень застосовано евристичний алгоритм табу-пошуку, який ефективно уникнув локальних мінімумів у просторі допустимих маршрутів. Для реалістичного моделювання трафіку використано фрактальний самоподібний неперервний пуасонівський процес (FSNDP), що адекватно відтворює статистичні властивості реального мережевого навантаження, зокрема довготривалу залежність та сплескову інтенсивність. Результати імітаційного експерименту продемонстрували переваги запропонованого підходу: зменшення середнього джиттера на 22–35 %, підвищення пропускної здатності на 18–27 % та покращення стійкості до тимчасових перевантажень у порівнянні з класичними протоколами, зокрема IEEE 802.11 CSMA/CA. Додатково реалізовано механізм активного контролю завантаження вузлів та відкидання надмірно повторюваних пакетів, що запобігає лавинному зростанню черг і знижує ймовірність колапсу мережі під час пікових навантажень.

Ключові слова: інтенсивність трафіку, якість обслуговування (QoS), бездротова мережа, модель маршрутизації, самоподібний трафік, релаксація Лагранжа, табу-пошук.