Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Кваліфікаційна
наукова праця на
правах рукопису

**КОВАЛЕНКО ОЛЕКСАНДР ВОЛОДИМИРОВИЧ**

УДК 004.05 (0.43.3)

**ДИСЕРТАЦІЯ**

# МОДЕЛІ ТА МЕТОДИ РОЗРОБЛЕННЯ БЕЗПЕЧНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

05.13.05 – Комп'ютерні системи та компоненти

Подається на здобуття наукового ступеня доктора технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ О.В. Коваленко

Науковий консультант: **Смірнов Олексій Анатолійович**
доктор технічних наук, професор

Черкаси – 2020

# АНОТАЦІЯ

*Коваленко О.В.* Моделі та методи розроблення безпечного програмного забезпечення комп’ютерних систем. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.05 «Комп'ютерні системи та компоненти». – Черкаський державний технологічний університет, Черкаси, 2020.

Дисертаційна робота спрямована на вирішення актуальної науково-технічної проблеми, що полягає в синтезі моделей та методів розроблення безпечного ПЗ КС.

У роботі проведено аналіз сучасних тенденцій розвитку методологій розроблення програмного забезпечення і вимог до програмних засобів, показників і критеріїв оптимізації, а також підходів математичної формалізації відповідних інформаційних процесів який показав, що в умовах впровадження комп'ютерних технологій в системи критичного застосування, збільшення інформації, що зберігається, обробляється і циркулює в них, а також підвищеної вразливості несанкціонованого доступу до ПЗ з боку зловмисників, використовувані нині моделі та методи розроблення ПЗ КС не дозволяють забезпечити необхідний рівень безпеки даних. На основі проведеного аналізу і міжнародних та державних стандартів сформовано загальну схему характеристик і показників, що відносяться до якості програмного забезпечення. Аналіз методологій розроблення програмного забезпечення і факторів, що впливають на безпеку, дозволив виділити протиріччя між підвищеними вимогами до безпеки ПЗ (врахуванням усіх вразливостей безпеки) і необхідністю адаптації до існуючих об'єктивно-суб'єктивних факторів, властивих сучасному світу ІТ-індустрії. Проведені порівняльні дослідження основних підходів математичної формалізації

дозволили визначити основні напрями дисертаційного дослідження і сформулювати оптимізаційне завдання синтезу моделей та методів розроблення ПЗ.

Удосконалено метод якісного аналізу вразливостей розроблення програмного забезпечення, що відрізняється від відомих врахуванням факторів експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки ПЗ КС, і оцінкою довільного несуперечливого кінцевого набору "квантів інформації". В основу синтезованого методу покладена структурна ідентифікація вразливостей розроблення ПЗ, що відрізняється від відомих побудовою оцінки вразливостей розроблення ПЗ «зверху» у вигляді множини, за наявності довільного несуперечливого кінцевого набору "квантів інформації". Це дозволило до 55% звузити сукупність множин Парето та більш точно обирати пріоритетністьність напрямків фінансування профілактичних заходів.

Удосконалено метод кількісної оцінки вразливостей розроблення ПЗ. Його відмінною особливістю є комплексне використання "Аналізу дерева відмов" і способу оцінки показника чистої приведеної вартості проекту розроблення ПЗ з урахуванням негативних факторів можливого невиявлення загроз безпеки ПЗ. Використання вдосконаленого "Аналізу дерева відмов" дозволить до 20% підвищити точність кількісної оцінки вразливостей розроблення ПЗ. В той же час, використання способу оцінки показника чистої приведеної вартості проекту розроблення ПЗ дозволяє розглядати проект комплексно, з урахуванням необхідності врахування безпеки і тестування вразливості ПЗ, із залученням інструментів, які дозволяють здолати складність, невизначеність і довгостроковість проектів.

Удосконалено метод оптимізації розподілу ресурсів розроблення безпечного ПЗ. В основу цього методу було покладено напівмарківську модель прийняття рішень для керованого марківського процесу у

безперервному часі. Відмінною особливістю запропонованого методу є використання псевдобулевих методів бівалентного програмування з нелінійною цільовою функцією і лінійними обмеженнями для визначення оптимальної стратегії усунення експлуатаційних помилок. Це дозволило оптимізувати процес проектування стратегії управління вразливостями.

Розроблено математичну модель технології тестування комплексу *DOM XSS* вразливостей, яка відрізняється від відомих урахуванням специфіки комплексного аналізу різних типів *XSS* вразливості ("*stored XSS*", "*reflected XSS*" і *DOM Based XSS*), а також включенням в алгоритм процедур автоматичного аудиту *DOM Based XSS* окремо. Це надало можливість провести аналітичну оцінку часових витрат тестування вказаних вразливостей в умовах реалізації стратегії розроблення безпечного програмного забезпечення.

Удосконалено математичну модель технології тестування вразливості до *SQL*-ін'єкцій, яка відрізняється від відомих вдосконаленим способом визначення відстані між результатами ін'єкції. Використання в запропонованому методі критерію Джаро-Вінклера для порівняння результатів ін'єкції *SQL*-коду і введення порогового значення дозволить підвищити точність результатів тестування безпеки ПЗ.

Розроблено метод математичного моделювання технологій тестування *DOM XSS* вразливості і вразливості до *SQL*-ін'єкцій, в основу якого покладено підхід мережевого *GERT* моделювання. Це дозволить досліджувати процеси в комп'ютеризованих системах при розробці нових засобів і протоколів захисту даних, а також від 1,05 до 1,5 разів зменшити час тестування безпеки.

Отримано подальший розвиток імітаційної моделі технології тестування безпеки на основі положень теорії масштабування імітаційних моделей. Відмінною особливістю розробленої імітаційної моделі є адаптація

вибору вхідних операторів управління і даних до підвищення вимог оперативності розроблення і реалізації моделі, виражена в реалізації процедури взаємодії з реальним браузером з використанням засобів автоматизації браузеру і формуванні даних для атаки на декількох діалектах. Це дозволило знизити обчислювальну складність реалізованих алгоритмів до 1,5 разів.

Отримано подальший розвиток методу передтестової компіляції і розподілу доступу, що відрізняється від відомих врахуванням профілів користувача при синтезі застосунку, а також використанням ресурсів "хмарних сховищ" в процесі отримання інсталяційних версій ПЗ. Це дозволило підвищити рівень безпеки розроблених застосунків.

Проведено порівняльну оцінку ефективності застосування розроблених моделей та методів, а також достовірності отриманих результатів. В цілому проведені дослідження показали, що показник безпеки ПЗ КС збільшився до 15%, що дозволяє зробити висновок про підвищення рівня захисту інформації за допомогою синтезованих моделей та методів розроблення безпечного ПЗ.

Після проведених експериментів результат показав, що статистична величина довірчої ймовірність відхилення для видів даних що розглядаються складає $P \approx 0{,}92$, тобто значення статистичної величини від математичного сподівання «не відхилиться» більше, ніж на одиницю. Результати дисертаційної роботи впроваджено в діяльність комерційних підприємств та навчальних закладах України.

**Ключові слова:** безпечне програмного забезпечення, ідентифікація вразливостей, якісний та кількісний аналіз вразливостей, безпека даних, оптимізація розподілу ресурсів розроблення програмного забезпечення, алгоритми тестування безпеки, масштабування, імітаційна модель, GERT-мережі.

## SUMMARY

*Kovalenko O.* Models and methods for developing secure software for computer systems. - Qualifying scientific work on the rights of the manuscript.

The dissertation on competition of a scientific degree of the doctor of technical sciences on a specialty 05.13.05 "Computer systems and components". - Cherkasy State Technological University, Cherkasy, 2020.

The dissertation is aimed at solving the current scientific and technical problem of development, improvement and selection of methods and models that provide maximum software security.

The paper analyzes the current trends in software development methodologies and software requirements, indicators and optimization criteria, as well as approaches to mathematical formalization of relevant information processes, which showed that in the introduction of computer technology in critical applications, increasing information that stored, processed and circulated in them, as well as the increased vulnerability of unauthorized access to software by attackers, currently used models and methods of software development of the COP do not allow to ensure the required level of data security. On the basis of the conducted analysis and the international and state standards the general scheme of the characteristics and indicators concerning quality of the software is formed. The analysis of software development methodologies and factors influencing security allowed to identify contradictions between increased software security requirements (taking into account all security vulnerabilities) and the need to adapt to existing objective and subjective factors inherent in the modern world of the IT industry. The conducted comparative researches of the basic approaches of mathematical formalization allowed to define the basic directions of dissertation research and to formulate the optimization problem of synthesis of models and methods of software development.

The method of qualitative analysis of software development vulnerabilities has been improved, which differs from the known ones by taking into account operational vulnerabilities, especially vulnerabilities of non-detection of security threats to CS software, and estimation of arbitrary consistent finite set of "information quanta". The synthesized method is based on the method of structural identification of software development vulnerabilities, which differs from the known by constructing an assessment of software development vulnerabilities "from above" in the form of a set, in the presence of an arbitrary consistent finite set of "information quanta". This allowed to narrow the set of Pareto sets to 55% and more precisely to choose the priority of areas of funding for preventive measures.

The method of quantitative assessment of software development vulnerabilities has been improved. Its distinctive feature is the integrated use of the method of "Failure Tree Analysis" and the method of estimating the net present value of the software development project, taking into account the negative factors of possible non-detection of software security threats. The use of an improved "Failure Tree Analysis" methodology will increase the accuracy of quantifying vulnerabilities in software development by up to 20%. At the same time, the use of the method of estimating the net present value of the software development project allows to consider the project comprehensively, taking into account the need to take into account security and vulnerability testing, using tools to overcome the complexity, uncertainty and long-term projects.

The method of optimizing the allocation of resources for the development of secure software has been improved. This method was based on the semi-Markov model of decision-making for a controlled Markov process in continuous time. A distinctive feature of the proposed method is the use of pseudo-Boolean methods of bivalent programming with nonlinear objective function and linear constraints to

determine the optimal strategy for eliminating operational errors. This allowed to optimize the process of designing a vulnerability management strategy.

A mathematical model of DOM XSS complex vulnerability testing technology has been developed, which differs from the known ones taking into account the specifics of complex analysis of different types of XSS vulnerabilities ("stored XSS", "reflected XSS" and DOM Based XSS), as well as inclusion of DOM Based XSS automatic audit procedures separately . This provided an opportunity to conduct an analytical assessment of the time spent testing these vulnerabilities in terms of implementing a strategy for developing secure software.

An improved mathematical model of technology for testing vulnerability to SQL injection, which differs from the known advanced method of determining the distance between the results of the injection. The use of the Jaro-Winkler test in the proposed method to compare the results of the injection of SQL code and the introduction of a threshold value will improve the accuracy of the results of software security testing.

A method of mathematical modeling of technologies for testing DOM XSS vulnerabilities and vulnerabilities to SQL-injections has been developed, which is based on the network GERT modeling approach. This will allow to study the processes in computer systems in the development of new tools and protocols for data protection, as well as from 1.05 to 1.5 times to reduce the time of security testing.

Further development of the simulation model of security testing technology based on the provisions of the theory of scaling of simulation models is obtained. A distinctive feature of the developed simulation model is the adaptation of the choice of input control operators and data to increase the efficiency of model development and implementation, expressed in the implementation of the procedure of interaction with a real browser using browser automation and data

generation to attack multiple dialects. This allowed to reduce the computational complexity of the implemented algorithms to 1.5 times.

Further development of the method of pre-test compilation and distribution of access, which differs from the known ones by taking into account user profiles in the synthesis of the application, as well as the use of "cloud storage" resources in the process of obtaining installation versions of software. This allowed to increase the level of security of the developed applications.

A comparative assessment of the effectiveness of the developed models and methods, as well as the reliability of the results. In general, studies have shown that the security index of the software of the COP has increased to 15%, which allows us to conclude that the level of information protection is increased using synthesized models and methods of developing secure software. After the experiments, the result showed that the statistical value of the confidence interval for the types of data under consideration is, ie the value of the statistical value from the mathematical expectation "will not deviate" by more than one. The results of the dissertation are implemented in the activities of commercial enterprises and educational institutions of Ukraine.

**Keywords:** secure software development, vulnerability identification, qualitative and quantitative analysis of vulnerabilities, data security, optimization of software development resource allocation, security testing algorithms, scaling, simulation model, GERT-networks.

## Список публікацій здобувача

[1] Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe **(Scopus)**

[2] Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». *CEUR Workshop Proceedings* Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe **(Scopus)**

[3] Kovalenko Oleksandr Qualitative risk analysis of software development / Oleksandr Kovalenko, Jamil Al-Azzeh, Oleksii Smirnov, Anna Kovalenko, Serhii Smirnov // Asian Journal of Information Technology. – Volume 17 Issue 3. – Medwell Journals. – 2018. – P. 218-230. ISSN: 1682-3915. URL: http://medwelljournals.com/abstract/?doi=ajit.2018.218.230 Doi: ajit.2018.218.230

[4] Kovalenko Oleksandr, The mathematical model of the testing technology for Dom XSS vulnerabilities / O. Kovalenko, O. Smirnov, A.Kovalenko, S. Smirnov, V. Vialkova // Scientific & practical cyber security journal (SPCSJ) Volume 2 Issue 1, P. 22-28. Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2018 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/04-3-o.kovalenko-a.kovalenko-o.smirnov-s.smirnov-v.vialkova.pdf

[5] Коваленко А.В. Технология тестирования DOM XSS

уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific & practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

[6] Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп'ютерних систем для підвищення безпеки даних: монография / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 350 с.

[7] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

[8] Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

[9] Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

[10] Коваленко А.В. Задачи распознавания ситуаций в ERP системах/ А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

[11] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 5(142). – Х.: ХУПС – 2016. – С. 153-157.

[12] Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". – Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

[13] Коваленко А.В. Метод качественного анализа рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

[14] Коваленко А.В. Метод количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

[15] Коваленко А.В. Использование псевдобулевых методов бивалентного программирования для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

[16] Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 2 (38). – Полтава: ПолтНТУ. – 2016. – С. 93-100.

[17] Коваленко А.В. Технология тестирования уязвимости к SQL инъекциям / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 5 (45). – Полтава: ПолтНТУ. – 2017. – С. 66-71.

[18] Коваленко А.В. Масштабирование имитационной модели

технологии тестирования безопасности / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (46). – Полтава: ПолтНТУ. – 2017. – С. 181-184.

[19] Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

[20] Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розроблення програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

[21] Коваленко О.В. Управління ризиками розроблення програмного забезпечення за умови обмеженості коштів виділених на усунення помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький: ЦНТУ. - 2018. – С. 128-140.

[22] Коваленко О.В. GERT-мережевий синтез технології тестування на вразливість WEB-додатків/ О.В. Коваленко // Захист інформації. – Випуск 20(2) – К.: НАУ. – 2018. – С. 89-94.

[23] Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

[24] Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141.

[25] Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному

транспорті. №4, 2018. – С. 41-44.

[26] Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

[27] Коваленко О.В. Аналіз та дослідження інформаційних технологій розробки програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

[28] Коваленко О.В. Удосконалений метод управління ризиками розроблення програмного забезпечення на основі напівмарковської моделі прийняття рішень/ О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2 (3 ). – Харків. – 2018. – С. 48-41.

[29] Коваленко О.В. Математичні моделі технології тестування DOM XSS вразливості та вразливості до SQL ін'єкцій / О.В. Коваленко // Вісник Черкаського державного технологічного університету. Серія : Технічні науки №4, 2018. – С. 29-36.

[30] Коваленко О.В. Математична модель технології тестування вразливості до SQL ін'єкцій / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (58). – Полтава: ПолтНТУ. – 2019. – С. 43-47.

[31] Коваленко О.В. Математична модель технології тестування комплексу DOM XSS вразливостей для аналітичної оцінки часових витрат / О.В. Коваленко // Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 173-180, 2019.

[32] Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.А. Смирнов, А.В. Коваленко // Збірник наукових праць II міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації».

м. Київ. 24-27 лютого 2016 р. – Київ: Європейський університет. – 2016. – С. 138-139.

[33] Коваленко А.В. Анализ и оценка рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез «Securitea informationala 2015-2016». Conferenta internationala (editia a XII-a). Chisinau. Moldova. 3 martie 2016. – Chisinau: ADSEM. – 2016. – P. 96-102.

[34] Коваленко А.В. Исследование источников и причин риска разработки программного обеспечения, этапов и работ, при выполнении которых возникает риск / А.А. Смирнов, А.В. Коваленко // Збірник тез VII всеукраїнської науково-практичної конференції "Інформатика та системні науки (ІСН-2016)". м. Полтава. 10-12 березня 2016 р. – Полтава.: ПУЕТ – 2016. – С. 264-266.

[35] Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

[36] Коваленко А.В. Методика структурной идентификации рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 24-25 березня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 71-72.

[37] Коваленко А.В. Методы качественного анализа рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016). м. Харків. 30 березня – 1

квітня 2016 р. – Харків: НТУ «ХПІ». – 2016. – С. 6-7.

[38] Коваленко А.В. Структурная идентификация рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 15-16 квітня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 175-182.

[39] Коваленко А.В. Исследование разработанной методики структурной идентификации рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез VIII міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 28-29 квітня 2016 р. – Харків: ХНЕУ. – 2016. – С. 49.

[40] Коваленко А.В. Исследование дерева рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез III міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХНМІ ДВНЗ «УБС». – 2016. – С. 174-178.

[41] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Сборник тезисов XII международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 30 мая – 02 июня 2016 г – Варна. ТУВ. – 2016. – С. 585-589.

[42] Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Кібербезпека в Україні: правові та організаційні питання». м. Одеса, 21 жовтня 2016 р. – Одеса : ОДУВС, 2016. – С.146-148.

[43] Коваленко А.В. Метод управления рисками разработки

программного обеспечения с использованием псевдобулевых методов бивалентного программирования / А.А. Смирнов, А.В. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Актуальні задачі та досягнення у галузі кібербезпеки». м. Кропивницький, 23-25 листопада 2016 року – Кропивницький: ЦНТУ, 2016. – С. 162.

[44] Коваленко А.В. Псевдобулевые методы бивалентного программирования для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко, С.А. Смирнов // Збірник наукових праць III міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 22-25 лютого 2017 р. – Київ: Європейський університет. – 2017. – С. 158-162.

[45] Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез II науково-практичної конференції "Проблеми кібербезпеки інформаційно-телекомунікаційних систем". м. Київ. 23-24 березня 2017 р. – Київ: КНУ ім. Тараса Шевченко – 2017. – С. 203-205.

[46] Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Conferenta internationala (editia a XIII-a). «Securitea informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

[47] Коваленко А.В. Алгоритм анализа DOM XSS уязвимости при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез дев'ятнадцятого міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кропивницький 7-8 квітня 2017 р. – Кропивницький: ГЛА НАУ. – 2017. – С. 125-127.

[48] Коваленко А.В. Алгоритм анализа уязвимости SQL Injection для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез другої міжнародної науково-технічної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2017). м. Харків.10-12 квітня 2017 р. – Харків: НТУ «ХПІ». – 2017. – С. 27.

[49] Коваленко А.В. Метод управления рисками разработки программного обеспечения на основе алгоритмов анализа уязвимостей / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 20-22 квітня 2017 р. – Кіровоград: КНТУ. – 2017. – С. 92.

[50] Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез IX міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. – 2017. – С. 61.

[51] Kovalenko O.V. Method of testing the DOM XSS vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii // International Conference «information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. – 2017. – P. 7.

[52] Коваленко О.В. Метод тестування DOM XSS уразливості / О.В. Коваленко, О.А. Смірнов, А.С. Коваленко, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості,

телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

[53] Коваленко О.В. GERT-модель технології тестування DOM XSS уразливості / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць IV міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 21-24 лютого 2018 р. – Київ: Європейський університет. – 2018. – С. 65-70.

[54] Коваленко О.В. Технології тестування уразливостей Web-застосунків з використанням GERT-моделі / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". м. Кривий Ріг. 21-23 березня 2018 р. – Кривий Ріг.: ДВНЗ КНУ – 2018. – С. 227-230.

[55] Коваленко А.В. Тестирование уязвимости Web-приложений к атаке вида межсайтовый скриптинг / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов, // Збірник тез «Securitea informationala 2018». Conferenta internationala (editia a XIV-a). Chisinau. Moldova. 20-21 martie 2018. – Chisinau: ADSEM. – 2018. – P. 54-56.

[56] Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов, // Збірник тез X міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

[57] Коваленко О.В. Розробка методу передтестової компіляції й розподілу доступу / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”,

м. Кропивницькй. 19-20 квітня 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215.

[58] Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL-ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

[59] Коваленко О.В. Аналіз основних підходів математичного моделювання та методологій для забезпечення максимальних показників безпеки програмного забезпечення / О.В. Коваленко, А.С. Коваленко // Збірник наукових праць всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених «Комп’ютерна інженерія і кібербезпека : досягнення та інновації, м. Кропивницькй. 27-29 листопада 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 74.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення;

КС – комп’ютерна система;

МПР – менеджер, який приймає рішення;

КСУ – комп’ютерна система управління;

ІП – інформаційний поток;

АП – авіатранспортне підприємство;

GERT – Graphical Evaluation and Review Technique;

ERP – Enterprise Resource Planning;

QA – Quality Assurance;

XP – Extreme Programming;

FTA – Fault Tree Analysis;

XSS – Cross-Site Scripting;

SIL – Safety Integrity Level;

NPV – Net Present Value;

OWASP – Open Web Application Security Project;

LLVM – Low Level Virtual Machine;

JIT – Just-in-time compilation;

SQL – Structured Query Language.

# ВСТУП

В теперішній час комп'ютерні системи (КС) є оснопологаючою ланкою існування сучасного суспільства та держави. Всі сфери життєдіяльності суспільства беспосередньо залежіть від якості та безпеки реалізації цих систем. При цьому однією з основних складових цих систем є програмне забезпечення, що виконує функції автоматизації, комунікації, управління, інтелектуалізації та ін.

Сучасні КС в даний час вирішують найвідповідальніші завдання різних галузей, при цьому чим складніше вирішувана завдання, тим більш критичними стають вимоги до компонентів КС.

Практичний досвід останніх років свідчить, що сучасні засоби та компоненти захисту програмного забезпечення комп'ютерних систем (ПЗ КС) не можуть забезпечити необхідний рівень безпеки. Почастішали випадки кібератак на комп'ютерні системи державних об'єктів критичної інфраструктури, на банківські рахунки, інформаційні ресурси оборонного відомства та інших державних служб. В першу чергу це пов'язано з тим, що з одного боку масове поширення комп'ютерних інформаційних мережевих технологій істотно розширило можливості та арсенал кіберзлочинців, а з іншого боку фірми-розробники найчастіше нехтують питаннями безпеки даних користувачів ПЗ. Крім того, рівень розвитку методологій розроблення програмного забезпечення не дозволяє акцентовано забезпечити ІТ-фірми необхідним методологічним і практичним контентом, що підвищує рівень безпеки.

Сучасний прогрес в області інформаційних технологій привів до розроблення численних методологій розроблення програмного забезпечення, призначених для забезпечення його якісних показників. Проте в додатку до

КС ці методи та засоби більшою мірою непридатні, тому що часто не мають належної теоретично усвідомленої основи, переваги та недоліки що розглядаються з описом властивостей системи спираються лише на локальний практичний досвід використання, що не є гарантом їх успішної роботи в умовах широкого спектру зловмисних дій та кібератак на ПЗ користувачів.

В теорії забезпечення кібербезпеки та захисту інформації накопичено значний теоретичний матеріал і практичний досвід. Найбільш суттєвими роботами в цій області є дослідження іноземних та вітчизняних учених, серед яких І. Д. Горбенко, В. І. Долгов, О. О. Кузнєцов, О. Г. Корченко, М. Брантон-Сполл, Д. Деннінг, Е. Спаффорд, Р. Смит, В. Столлінгс, Б. Шнайєр Р. С. Сиакорд та ін. Але враховуючі динамічний розвиток інтелектуалізації комп'ютеризованих керуючих рішень, сучасні інформаційні технології, різноманітність технологічних та апаратно-інформаційних рішень сучасного підходу програмування сприяють тому, що постановка завдань підвищення безпеки ПЗ КС істотно видозмінюється це пов'язано з тим, що необхідно враховувати дії нових факторів, до яких можна віднести наступні:

– ПЗ КС є елементом підвищеного інтересу зловмисників, що істотно підвищує рівень і різноманітність зовнішніх впливів;

– в розробці ПЗ КС бере участь багато фахівців різного напряму (кодери, дизайнери, тестувальники та ін.), що істотно ускладнює процес управління, а також вимагає додаткових заходів з підвищення безпеки інформаційних складових системи;

– саме по собі ПЗ є критичним - у міру того, що ПЗ все частіше використовується, його збої можуть впливати на все більшу кількість об'єктів і суб'єктів;

– існує широкий спектр зловмисного програмного забезпечення, що має можливості реверсних заходів, спрямованих на деструктивні зміни легального ПЗ.

У зв'язку з цим, особливої актуальності набувають питання синтезу моделей та методів розроблення безпечного ПЗ КС. При цьому їх ключовою особливістю є врахування і адаптація існуючих гнучких методологій розроблення ПЗ до підвищених вимог безпеки КС з використанням сучасного математичного, методологічного і технологічного апарату ІТ-фірм. Врахування перерахованих особливостей ПЗ КС виходить за рамки існуючих моделей та методів розроблення ПЗ і вимагає як модифікації, так і їх перегляду.

Таким чином, одночасно зі збільшенням кількості ПЗ КС, підвищенням вимог до його безпеки відбувається збільшення кількісного і підвищення якісного рівня зловмисних дій, що призводить до протиріччя між зниженням показників безпеки ПЗ КС, викликаним зовнішніми зловмисними діями з одного боку, і жорсткими вимогами до гарантованого рівня безпеки даних з іншого боку.

Усе вищезазначене визначає **актуальність** нових моделей та методів розроблення безпечного ПЗ КС.

При одночасній наявності потреби в змінах як процесу розроблення ПЗ, так і самого ПЗ, а також вразливості, що неявно виникає при зміні ПЗ КС, для дослідників виявляється найбільш важливою розроблення моделей та методів, що дозволяють успішно виконувати процес безпечної еволюції ПЗ. При цьому кінцевою метою дисертаційної роботи є виявлення недоліків процесу розроблення ПЗ, причин уразливості програмних засобів, і розроблення відповідних моделей, методів і засобів, що дозволять підвищити безпеку даних.

Врахування основних вразливостей розроблення ПЗ, управління вразливостями в процесі розроблення та експлуатації, використання спеціально розробленої моделі тестування найбільш небезпечних вразливостей, а також методів передтестової компіляції і розподілу доступу, дозволяє зменшити час і підвищити точність кількісної оцінки вразливостей розроблення ПЗ, зменшити час тестування безпеки, тобто підвищити рівень безпеки даних.

На основі виконаних теоретичних і експериментальних досліджень, в дисертаційній роботі сформульована і вирішена важлива для теорії і практики **науково-технічна проблема**, що полягає в синтезі моделей та методів розроблення безпечного ПЗ КС.

**Зв'язок роботи з науковими програмами, планами, темами.** Дослідження в дисертаційній роботі проводилися відповідно до наступних нормативних актів:

– Концепцією Національної Програми інформатизації, схваленої Законом України «Про Концепцію Національної програми інформатизації» від 04.02.1998 р. № 75\98 - ВР (зі змінами 2000 - 2012 рр.);

– Законом України «Про телекомунікації» від 18.11.2003 р. № 1280-IV.

– Законом України «Про захист інформації в інформаційно-телекомунікаційних системах» від 31.05.2005 р. № 2594-IV;

– Постановою Кабінету Міністрів України від 29.03.2006 р. №373 «Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах» (зі змінами 2006, 2011 рр.);

– планами наукової і науково-технічної діяльності Центральноукраїнського національного технічного університету, у рамках виконання науково-дослідних робіт: держбюджетних науково-дослідних робіт №36Б113 «Розробка методів підвищення оперативності передачі і

захисту інформації в телекомунікаційних системах» (ДР №0113U003086), №36Б115 «Розробка методів синтезу тестових моделей поведінки програмних об'єктів, підвищення оперативності передачі і захисту інформації в телекомунікаційних системах» (ДР №0115U003103), науково-дослідних робіт «Інформаційна технологія автоматизації проектування і тестування об'єктно-орієнтованого програмного забезпечення» (ДР №0114U003831), «Інформаційна технологія проектування тестових наборів на основі вимог до програмного забезпечення інфокомунікаційних систем» (ДР №0116U008133), в який автор є співвиконавцем окремих етапів.

**Мета і завдання дослідження.** Метою дисертаційної роботи є підвищення рівня безпеки програмних компонентів комп’ютерних систем шляхом створення нових та удосконалення існуючих моделей та методів розроблення безпечного програмного забезпечення.

Відповідно до поставленої мети для вирішення науково-технічної проблеми в дисертаційній роботі необхідно вирішити наступні взаємопов'язані завдання:

1. Удосконалення методу якісного аналізу вразливостей розроблення ПЗ, що враховує фактори експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки ПЗ КС.

2. Удосконалення методу кількісної оцінки вразливостей розроблення ПЗ, що враховує негативні фактори можливого невиявлення загроз безпеки ПЗ КС.

3. Удосконалення методу оптимізації розподілу ресурсів розроблення ПЗ з використанням напівмарківської моделі прийняття рішень для керованого марківського процесу у безперервному часі.

4. Розробка математичної моделі технології тестування комплексу *DOM XSS* вразливостей, що враховує специфіки комплексного аналізу різних типів *XSS* вразливості.

5. Розробка математичної моделі технології тестування вразливості до *SQL*-ін'єкцій на основі критерію Джаро-Вінклера.

6. Розробка методу математичного моделювання процесу тестування *DOM XSS* вразливості та вразливості до *SQL*-ін'єкцій на основі підходу мережевого *GERT* моделювання.

7. Удосконалення імітаційної моделі технології тестування безпеки на основі положень теорії масштабування імітаційних моделей з урахуванням адаптації вибору вхідних операторів управління і даних до підвищення вимог оперативності розроблення.

8. Розробка методу передтестової компіляції і розподілу доступу, що враховує профілі користувачів при розроблені застосунку, а також використання ресурсів «хмарних сховищ» в процесі отримання інсталяційних версій.

*Об'єкт дослідження* - процеси розроблення і експлуатації безпечних програмних компонентів комп'ютерних систем.

*Предмет дослідження* - моделі, методи та засоби розроблення безпечного програмного забезпечення комп'ютерних систем.

**Методи дослідження.** При вирішенні науково-технічної проблеми було використано широкий спектр методів. Так, при розробці математичних моделей процесу тестування *DOM XSS* вразливості та вразливості до *SQL*-ін'єкцій використовувалися методи теорії графів і мережевого *GERT* моделювання. При розробці методу якісного аналізу використовувався інструмент аналізу причинно-наслідкових зв'язків між різними факторами і вразливостями, а також звуження Парето за допомогою «кванта» інформації. Для кількісної оцінки вразливостей використовувалася графічна модель *FTA*. В основу методу оптимізації розподілу ресурсів розроблення ПЗ була покладена напівмарківська модель прийняття рішень для керованого марківського процесу у безперервному часі. Імітаційне моделювання було

проведене на основі положень теорії масштабування імітаційних моделей. Оцінка експериментальних даних, отриманих в ході роботи, проводилася на основі методів математичної статистики.

Вибір методів досліджень забезпечив достовірність отриманих результатів і висновків, що підтверджується збіжністю результатів експериментальних досліджень, отриманих при програмній реалізації алгоритмів тестування вразливості ПЗ КС, з теоретичними і практичними результатами, відображеними в публікаціях.

**Наукова новизна отриманих результатів.** В підсумку у результаті виконання дисертаційної роботи отримав подальший розвиток науковий напрям, пов'язаний з синтезом моделей та методів розроблення безпечного ПЗ КС. У рамках цього напряму отримані наступні наукові результати:

1. *Удосконалено* метод якісного аналізу вразливостей розроблення програмного забезпечення, що відрізняється від відомих врахуванням факторів експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки ПЗ КС, і оцінкою довільного несуперечливого кінцевого набору «квантів інформації». Це дозволило звузити множину важливих вразливостей і знизити можливі фінансові та іміджеві втрати організацій-розробників ПЗ.

2. *Удосконалено* метод кількісної оцінки вразливостей розроблення ПЗ, що відрізняється від відомих комплексним використанням «Аналізу дерева відмов» і способу оцінки показника чистої приведеної вартості проекту розроблення безпечного ПЗ з урахуванням негативних факторів можливого невиявлення загроз безпеки ПЗ КС. Це дозволило підвищити точність кількісної оцінки вразливостей розроблення ПЗ.

3. *Удосконалено* метод оптимізації розподілу ресурсів розроблення ПЗ на основі напівмарківської моделі прийняття рішень для керованого марківського процесу у безперервному часі. Відмінною особливістю

запропонованого методу є використання псевдобулевих методів бівалентного програмування з нелінійною цільовою функцією і лінійними обмеженнями для визначення оптимальної стратегії усунення експлуатаційних помилок. Це дозволило оптимізувати процес проектування стратегії розподілу ресурсів розроблення ПЗ.

4. *Вперше розроблено* математичну модель технології тестування комплексу *DOM XSS* вразливостей, яка відрізняється від відомих урахуванням специфіки комплексного аналізу різних типів *XSS* вразливості («*stored XSS*», «*reflected XSS*» і *DOM Based XSS*), а також включенням в алгоритм процедур автоматичного аудиту *DOM Based XSS* окремо. Це дозволило провести аналітичну оцінку часових витрат тестування вказаних вразливостей в умовах реалізації стратегії розроблення безпечного програмного забезпечення.

5. *Вперше розроблено* математичну модель технології тестування вразливості до *SQL*-ін'єкцій, яка відрізняється від відомих використанням критерію Джаро-Вінклера, для порівняння результатів ін'єкції *SQL*-коду і введення порогового значення. Це дозволило підвищити точність результатів тестування безпеки програмного забезпечення.

6. *Вперше розроблено* метод математичного моделювання технологій тестування *DOM XSS* вразливості та вразливості до *SQL*-ін'єкцій, в основу якої покладений підхід мережевого *GERT* моделювання. Це дозволило досліджувати процеси в комп'ютеризованих системах при розробці нових засобів і протоколів захисту даних, а також зменшити час тестування безпеки програмного забезпечення.

7. *Отримано подальший розвиток* імітаційної моделі технології тестування безпеки на основі положень теорії масштабування імітаційних моделей. Відмінною особливістю розробленої імітаційної моделі є адаптація вибору вхідних операторів управління і даних до підвищення вимог

оперативності розроблення і реалізації моделі, виражена в реалізації процедури взаємодії з реальним браузером, з використанням засобів автоматизації браузеру і формуванні даних для атаки на декількох діалектах. Це дозволило понизити обчислювальну складність алгоритмів, що реалізовуються.

8. *Вперше розроблено* методу передтестової компіляції і розподілу доступу, що відрізняється від відомих врахуванням профілів користувача при розроблені застосунку, а також використанням ресурсів «хмарних сховищ» в процесі отримання інсталяційних версій. Це дозволило підвищити рівень безпеки застосунків, що розробляються.

**Практичне значення отриманих результатів** в області розроблення безпечного ПЗ полягає в тому, що синтезовані в дисертаційній роботі моделі та методи є науково-методичною основою для розроблення відповідних сучасних алгоритмів, сервісів, системних компонентів, протоколів, кроссплатформенних програмних засобів інформаційного і методологічного забезпечення. Практична значущість отриманих результатів полягає в наступному:

– метод математичного моделювання технологій тестування *DOM XSS* вразливості та вразливості до *SQL*-ін'єкцій дозволив розробити автоматизований програмний засіб виявлення вразливості ПЗ;

– метод передтестової компіляції і розподілу доступу дозволив розробити відповідний програмний комплекс;

– метод кількісної оцінки вразливостей розроблення ПЗ дозволив спроектувати програмну систему оцінки чистої приведеної вартості проекту розроблення ПЗ.

Практична значущість отриманих результатів підтверджується їх застосуванням (Додаток А):

– при розробці автоматизованих систем виявлення вразливостей ПЗ Інтернет сервіс провайдері ТОВ «ІМПЕРІАЛ-НЕТ» (м. Кропивницький);

– удосконаленні гнучкої методології розроблення ПЗ у компанії-розробнику програмного забезпечення ТОВ «МІФ ПРОДЖЕКТС» (м. Кропивницький);

– при розробці систем захисту інформації ТОВ «САЙФЕР ІТ» (м. Київ);

– в учбовому процесі Центральноукраїнського національного технічного університету.

**Особистий вклад здобувача.** Усі основні наукові положення, результати, висновки і рекомендації, приведені в дисертаційній роботі, отримані автором самостійно. Вони викладені як в роботах, які опубліковані без співавторів [6, 16-31], так і у співавторстві. У наукових роботах, які опубліковані у співавторстві, внесок автора полягає в наступному: у [1] розроблено метод оцінки рівнів зрілості програмних систем та підходу розрахунку тривалості необхідного прогнозованого періоду; у [2] розроблено комплекс самоподібних генераторів трафіку за допомогою ланцюгів Маркова, які відрізняються від аналогів меншими вимогами до обчислювальної потужності для імітаційних систем технологій тестування безпеки; у [3] проаналізовано показники якості вразливостей розроблення програмного забезпечення; у [4] розроблено математичну модель технології тестування на вразливості *DOM XSS*; у [5] розроблено технологію тестування *DOM XSS* вразливості; у [7] проведено аналіз підходів розроблення безпечного програмного забезпечення з використанням запропонованих методів якісного аналізу та кількісної оцінки вразливостей; у [8] розроблено метод оптимізації розподілу ресурсів розроблення безпечного програмного забезпечення; у [9] розроблено комплекс математичних моделей технології тестування *web*-застосунків; у [10] сформульовано задачі розпізнавання ситуацій у *ERP*-системах; у [11] розроблено методи якісного аналізу та

кількісної оцінки вразливостей розроблення програмного забезпечення; у [12] визначено проблеми аналізу та оцінки вразливостей інформаційної діяльності; у [13] розроблено метод якісного аналізу вразливостей розроблення програмного забезпечення; у [14] розроблено метод кількісної оцінки вразливостей розроблення програмного забезпечення; у [15] обґрунтовано використання псевдобулевих методів бівалентного програмування для оптимізації розподілу ресурсів розроблення безпечного програмного забезпечення.

Зазначений особистий внесок здобувача в роботах, які виконані у співавторстві, відповідає темі та змісту дисертації.

**Апробація результатів дисертації.**

Основні положення дисертаційної роботи були представлені на наступних конференціях: «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації» (Київ, 2016-2018 рр.); «*Securitea informationala*» (*Chisinau, Moldova*, 2016-2018 рр.); «Інформатика та системні науки» (Полтава, 2016 р.); «Проблеми кібербезпеки інформаційно-телекомунікаційних систем» (Київ, 2016-2017 рр.); «Інформаційна безпека та комп'ютерні технології» (Кропивницький, 2016-2018 рр.); «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (Харків, 2016-2017 р.); «Комбінаторні конфігурації та їх застосування» (Кропивницький, 2016-2017 рр.); «Проблеми і перспективи розвитку IT-індустрії» (Харків, 2016-2018 рр.); «Інформаційна та економічна безпека» (Харків, 2016 р.); «Стратегия качества в промышленности и образовании» (Варна, Болгарія, 2016, 2018 рр.); «Кібербезпека в Україні: правові та організаційні питання» (Одеса, 2016 р.); «Актуальні задачі та досягнення у галузі кібербезпеки» (Кропивницький, 2016 р.); «*Information technologies, systems and networks*» (*Chisinau, Moldova*, 2017 р.); «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях,

енергетиці та транспорті» (Кропивницький, 2017 р.); «Комп'ютерні інтелектуальні системи та мережі» (Кривий Ріг, 2018 р.); «Комп’ютерна інженерія і кібербезпека: досягнення та інновації» (Кропивницький, 2018 р.).

**Публікації** Основні положення і результати дисертації опубліковано у 59 наукових працях, у тому числі: 1 монографія; 3 колективні монографії; 2 наукові статті у міжнародному рецензованому виданні, що входить до бази даних Scopus; 3 наукові статті у закордонних фахових наукових журналах, та 22 статті у наукових журналах та збірниках наукових праць, що входять до переліку фахових видань України ( з них без співавторів – опубліковано 16), а також 28 матеріалів і тез доповідей на всеукраїнських та міжнародних конференціях.

**Структура і об'єм дисертації.** Дисертація складається з анотації, змісту, вступу, шести розділів, висновків, списку використаних джерел і додатків. Загальний обсяг роботи становить 305 сторінки, з них обсяг основного тексту – 250 сторінок, 58 рисунків, 17 таблиця, список використаних джерел складає 269 найменувань і займає 28 сторінок, а також 3 додатки на 27 сторінках.

# РОЗДІЛ 1
# АНАЛІЗ І ДОСЛІДЖЕННЯ МОДЕЛЕЙ ТА МЕТОДІВ РОЗРОБЛЕННЯ БЕЗПЕЧНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. ОБҐРУНТУВАННЯ ВИБОРУ НАПРЯМУ ДОСЛІДЖЕННЯ

У даному розділі проводиться аналіз основних тенденцій розвитку моделей та методів розроблення безпечного програмного забезпечення і вимог до програмних засобів. Наведено результати досліджень сучасних моделей та методів розроблення безпечного програмного забезпечення і факторів, що впливають на безпеку. Проведено аналіз і порівняльне дослідження основних підходів математичного моделювання процесу розроблення безпечного програмного забезпечення. На основі виявлених закономірностей, переваг і недоліків сучасних методологій розроблення безпечного програмного забезпечення здійснюється постановка завдання дисертаційного дослідження.

## 1.1 Аналіз сучасних тенденцій розвитку моделей і методів розроблення безпечного програмного забезпечення, а також вимог до програмних засобів

Сучасний період розвитку засобів автоматизації та інформатизації суспільства в цілому і окремих організацій та підприємств зокрема можна охарактеризувати як час масового переходу від стихійної комп'ютеризації окремих елементів діяльності організацій до єдиних інтегрованих рішень, що охоплюють усі аспекти їх існування.

Це не могло не відобразитися на складі і об'ємі найчастіше виконуваних IT-проектів і на методах їх виконання.

Особливостями сучасних IT-проектів є:

– замовлення (розроблення) і експлуатація уніфікованих програмних застосунків та *ERP*-систем;

– перехід до розділення праці в проектах з розроблення ПЗ;

– підвищення вимог до якості ПЗ;

– залучення замовника до процесу розроблення та ін.

Динамічний розвиток цифрових технологій, зростання кількості пристроїв, підключених до Інтернету речей, прийняття закону Сарбейнса-Окслі, зміна нормативно-правового середовища, часті фінансові кризи, відмови в роботі критично важливого устаткування, терористичні атаки і зростання кіберзлочинності - лише деякі з причин, які примушують суспільство вдосконалювати свої інформаційні системи і засоби захисту програмного забезпечення.

Особливо гострою ця проблематика виглядає в умовах використання комп'ютерних систем критичного застосування, вимоги до безпеки програмних засобів в яких надзвичайно високі.

В даний час українське законодавство регламентує питання безпеки програмного забезпечення на підставі закону "Про авторське право і суміжні права" в редакції від 11 червня 2001 року [1, 2], а також підзаконними актами, прийнятими Верховною Радою України (ДСТУ ISO/IEC 25010:2016, ДСТУ ISO/IEC 25012:2016, ДСТУ ISO/IEC 25021:2016 та ін.) [3-5], у яких під якістю програмного забезпечення розуміється набір ознак і властивостей програмних засобів, які характеризують здатність задовольняти встановлені і передбачені потреби.

Проте, відповідно до вказаних документів вибір метрик оцінки якості і відповідно до безпеки виконується суб'єктивно. Це значною мірою ускладнює процес об'єктивної оцінки якості програмного забезпечення в цілому і безпеці зокрема, що у свою чергу знижує цінності процедур сертифікації програмного забезпечення.

В той же час існує ряд робіт [6-12], у яких пропонується регламентувати цей процес і видаються можливі варіанти і шляхи вирішення цієї задачі.

Проведений аналіз і дослідження дозволили сформувати загальну схему характеристик і показників, що відносяться до якості програмного забезпечення, а також виділити серед них характеристики безпеки (рис. 1.1).

Як видно з рис. 1.1, якість ПЗ можна представити як функцію восьми складових: підфункцій придатності ($F_a$), ефективності ($F_e$), сумісності ($F_c$), зручності використання ($F_u$), супровідності ($F_{es}$), можливості перенесення і встановлення ($F_t$), функціональної безпеки ($F_r$), інформаційної безпеки ($F$).

При цьому характеристика придатності є функцією показників функціональної повноти ($F_f$), функціональної коректності ($F_{fcr}$), функціональної доцільності ($F_{fe}$).

Показники, що характеризують ефективність, можна представити у вигляді вектору, складовими елементами якого є функції поведінки в часі ($F_{bc}$), поведінки ресурсів ($F_{br}$), місткість ($F_{fca}$).

Сумісність ПЗ - це функція, показниками якої є співіснування ($F_{fco}$) і взаємодія ($F_{fi}$).

У свою чергу, характеристику "зручність використання" можна представити як сукупність показників доцільності ($F_{fre}$), керованості ($F_{con}$), захисту від помилок користувачів ($F_{um}$), естетичності інтерфейсу ($F_{fa}$), доступності користувачів ($F_{fa}$).

Супровідність є функцією показників модульності ($F_{fmod}$), можливості повторного використання ($F_{frei}$), можливості аналізу ($F_{fan}$), можливості модифікації ($F_{fmod}$), можливості тестування ($F_{ftes}$).

Можливість перенесення і встановлення, як ще одна важлива характеристика, може описуватися за допомогою показників адаптованості ($F_{fadp}$), можливості інсталяції ($F_{fins}$), можливості заміни ($F_{frep}$).

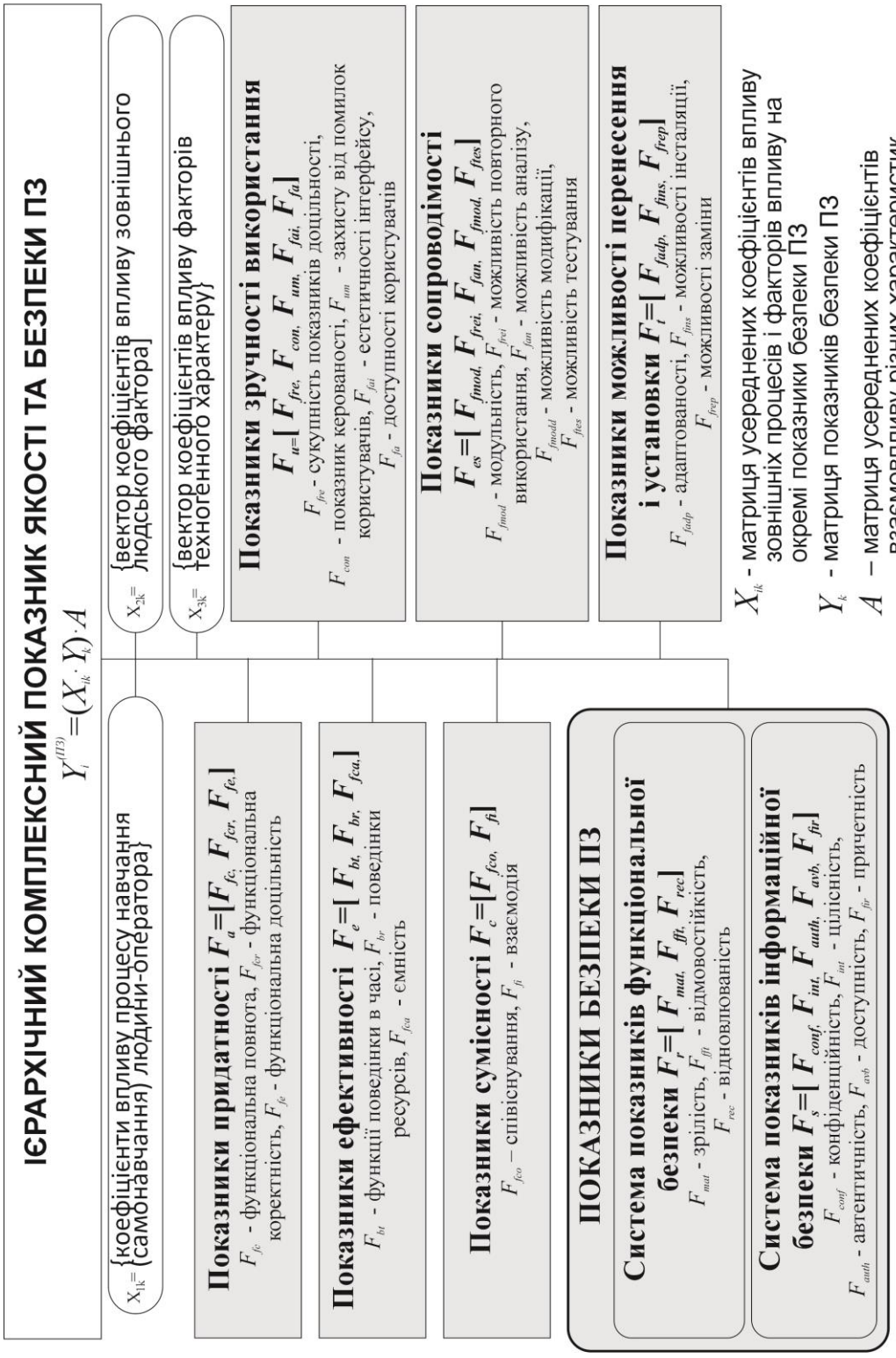


Рис. 1.1. Загальна схема характеристик і показників, що відносяться до якості програмного забезпечення

Дослідження показали, що групу характеристик безпеки ПЗ (функціональної безпеки ($F_r$), інформаційної безпеки ($F_s$)) доцільно

розглядати з точки зору єдиної мети - забезпечення безпеки комп'ютерних систем, а також існуючій можливості взаємовпливу один на одного. При цьому показниками функціональної безпеки є зрілість ($F_{mat}$), відмовостійкість ($F_{fft}$), відновлюваність ($F_{rec}$), а показниками інформаційної безпеки є конфіденційність ($F_{conf}$), цілісність ($F_{int}$), автентичність ($F_{auth}$), доступність ($F_{avb}$), причетність ($F_{fir}$).

У роботах [7, 10, 13-16] проводилися дослідження і були розроблені комплексні показники якості функціонування КС. Скористаємося системними підходами, представленими в цих роботах, і розробимо комплексний показник безпеки ПЗ КС.

Проведені дослідження і аналіз показників якості ПЗ у вигляді ієрархічної векторної системи (рис. 1.1.), а також показників безпеки ПЗ зокрема, дозволили представити комплексний показник безпеки ПЗ КС $\mathrm{Y}_{\mathrm{i}}^{(\mathrm{ПЗ})}$ у вигляді добутку матриць:

$$\mathrm{Y}_{\mathrm{i}}^{(\mathrm{ПЗ})} = (\mathrm{X}_{\mathrm{ik}} \cdot \mathrm{Y}_{\mathrm{k}}) \cdot \mathrm{A}, \qquad (1.1)$$

де $X_{ik} = \left[ x_{\psi}^{(\xi)} \right]$ – матриця усереднених коефіцієнтів впливу зовнішніх процесів і факторів впливу на окремі показники безпеки ПЗ, $i$ – кількість зовнішніх факторів, що впливають на функціонування системи, $k$ – кількість програмних засобів ПЗ КС, $x_{\xi}^{(\psi)} = \frac{1}{N} \sum_{j=1}^{N} x_{\ell_j}^{(\psi)}$ – усереднений коефіцієнт впливу зовнішніх процесів ($\psi$) на показники безпеки окремих програмних засобів КС ($\xi$), $\ell$ - найменування окремого показника безпеки ПЗ, $A$ – матриця усереднених коефіцієнтів взаємовпливу різних характеристик якості ПЗ, $\mathrm{Y}_{\mathrm{k}} = \left[ \overline{Y_{mat}^{(\mathrm{ПЗ})}}, \overline{Y_{fft}^{(\mathrm{ПЗ})}}, \overline{Y_{rec}^{(\mathrm{ПЗ})}}, \overline{Y_{conft}^{(\mathrm{ПЗ})}}, \overline{Y_{intt}^{(\mathrm{ПЗ})}}, \overline{Y_{autht}^{(\mathrm{ПЗ})}}, \overline{Y_{avb}^{(\mathrm{ПЗ})}}, \overline{Y_{fir}^{(\mathrm{ПЗ})}} \right]$ – матриця показників безпеки ПЗ, $\overline{Y_{mat}^{(\mathrm{ПЗ})}}, \overline{Y_{fft}^{(\mathrm{ПЗ})}}, \overline{Y_{rec}^{(\mathrm{ПЗ})}}, \overline{Y_{conft}^{(\mathrm{ПЗ})}}, \overline{Y_{intt}^{(\mathrm{ПЗ})}}, \overline{Y_{autht}^{(\mathrm{ПЗ})}}, \overline{Y_{avb}^{(\mathrm{ПЗ})}}, \overline{Y_{fir}^{(\mathrm{ПЗ})}}$ – векторні показники безпеки ПЗ.

В результаті перемноження (вираз 1.1) буде сформована матриця, що являє собою комплексний показник безпеки $Y_i^{(ПЗ)}$ ПЗ КС.

Надалі, в процесі моделювання і розроблення програмного забезпечення для підвищення безпеки ПЗ, застосування і аналізу її структурних елементів, для вирішення окремих оптимізаційних завдань доцільно виставляти прапор пріоритетності на окремі елементи матриць $X_{ik}, Y_k$ і $A$.

Таким чином, використання методів просторово-матричного представлення систем, а також єдиний підхід до конвергенції показників функціональної та інформаційної безпеки, дозволили розробити ієрархічний комплексний показник безпеки ПЗ КС, який враховує як параметри функціональної та інформаційної безпеки, так і фактор зовнішніх впливів. Виконання цього показника дає підставу вважати результатом розроблення безпечного ПЗ.

## 1.2 Аналіз основних моделей та методів розроблення безпечного програмного забезпечення і факторів, що впливають на безпеку

Аналіз вимог безпеки в законодавчих і регламентуючих актах, дослідження представленої вище моделі якості ПЗ, а також досвід розроблення, специфіки впровадження і супроводу ПЗ дозволили зробити висновок про значне підвищення вимог до безпеки, особливо в КС.

Відповідно до керівних документів [1, 3-5, 17-26], нині існує ряд способів забезпечення безпеки програмного забезпечення. Серед них можна виділити способи, що відносяться до обґрунтованої розроблення і ефективного виконання політики безпеки, оперативного реагування на події, що відносяться до безпеки ПЗ, безпечного програмування та ін.

Усі ці способи в тій чи іншій мірі повинні виконуватися відповідно до методології (керівництвом з безпеки програмного забезпечення) [1, 18, 19, 26, 27], що призводить до фізичного сенсу інтегрального рівня безпеки (*SIL - Safety Integrity Level*).

Відповідно до цієї методології, дії, пов'язані із забезпеченням безпеки програмного забезпечення, починаються на початкових стадіях роботи над проектом і тривають впродовж усього циклу розроблення, причому багато дій виконуються паралельно. На рис. 1.2. показано, як дії, пов'язані із забезпеченням безпеки програмного забезпечення, накладаються на різні інші дії, що виконуються в процесі його розроблення.

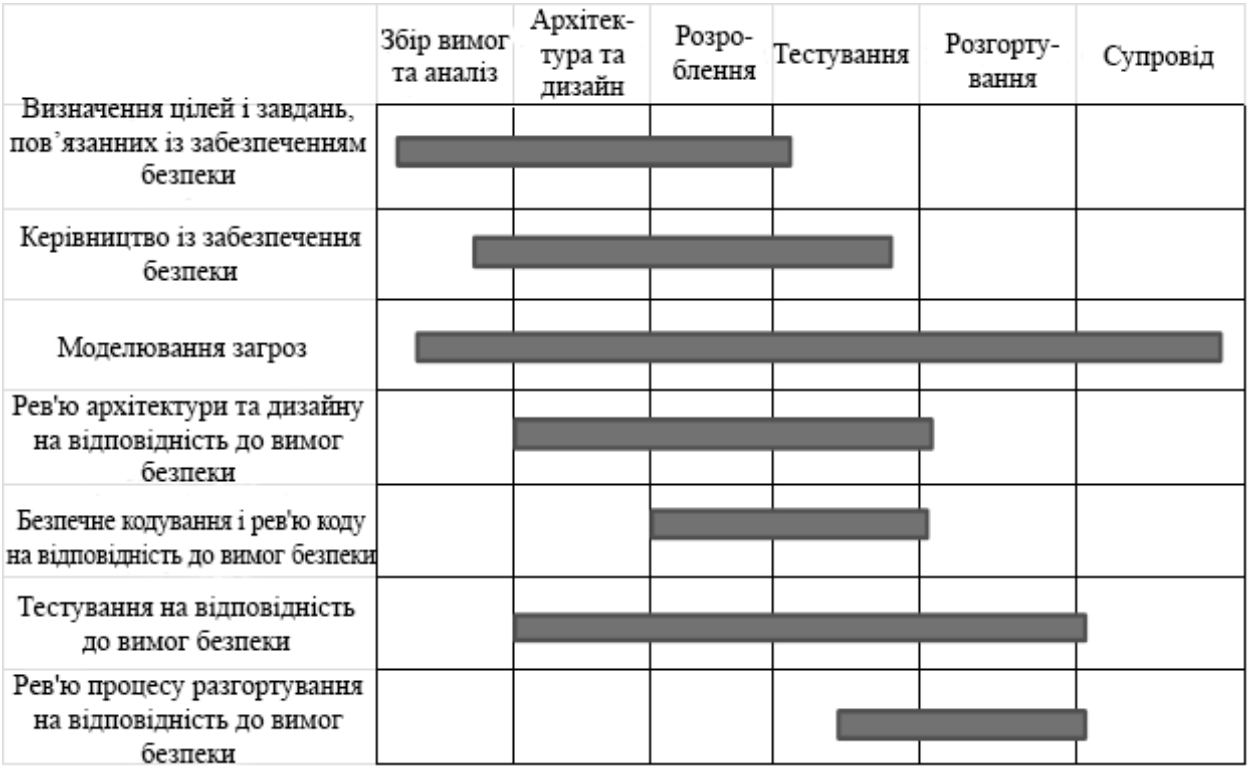


Рис. 1.2. Діаграма операцій забезпечення безпеки у циклі розроблення програмного забезпечення

Слід зауважити, що при розробці безпечного ПЗ дуже важливим кроком є обґрунтований вибір методології реалізації цього завдання [28-31],

що дозволяє підвищити ефективність і якість програмного продукту. При цьому необхідно враховувати, що сукупність процесів розроблення ПЗ є складною багатофакторною системою проходжень етапів у рамках вибраної методології. Аналіз літератури [32-36] показав, що сучасні методології розроблення ПЗ можна розділити на три основні групи: послідовні (прогнозовані), циклічні (напівпрогнозовані), гнучкі (абстрактні).

Основні відмінності між гнучкими і послідовними методологіями розроблення ПЗ представлені на рис. 1.3. При цьому слід зауважити, що філософія *Agile* включає різні принципи і правила, об'єднані між собою в один документ (маніфест *Agile*). На рис. 1.3. виділені тільки основні правила.

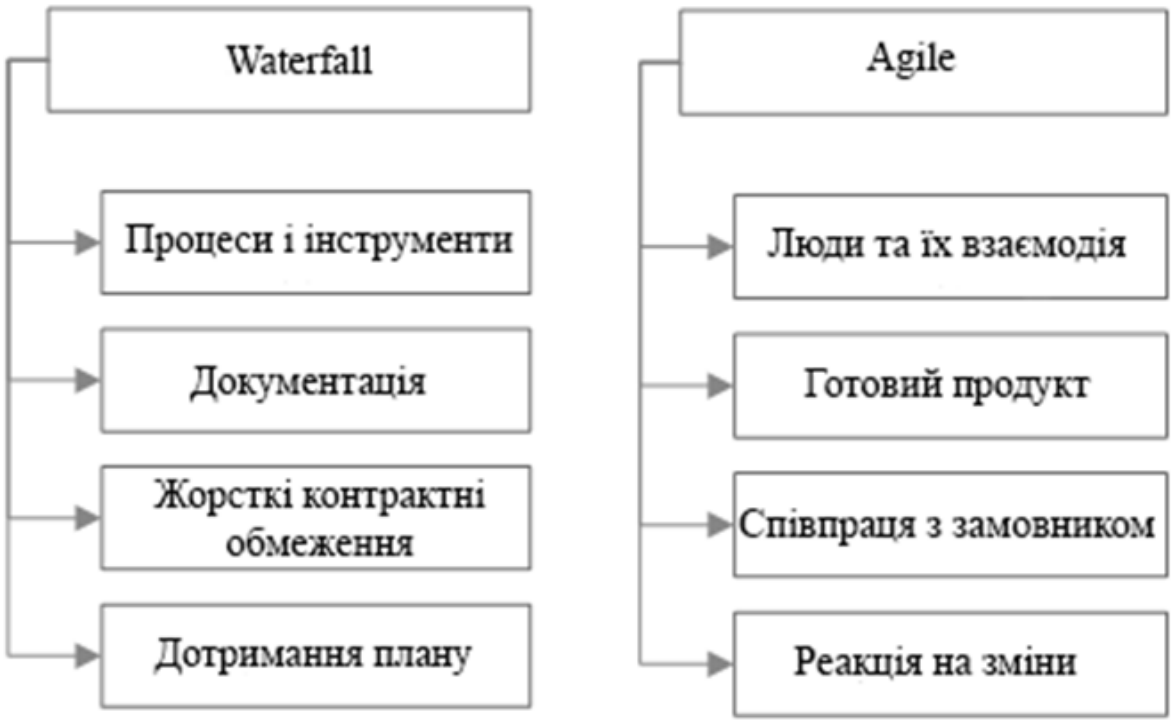


Рис. 1.3. Основні відмінності між гнучкими та послідовними методологіями

Порівняльне дослідження звісних методологій розроблення ПЗ дозволило визначитись з їх перевагами та недоліками стосовно питань безпеки [33, 35, 37-39]. Результати порівняння в загальному вигляді представлено в табл. 1.1.

Аналіз літератури, а також методичних рекомендацій з розроблення ПЗ у ряді джерел, дає підстави стверджувати про те, що при розробці безпечного ПЗ для КС нині переважно використовують послідовні (прогнозовані) методології. Серед них перевага віддається водоспадній моделі. Проте сучасні тенденції в розробці, а також багато об'єктивно-суб'єктивних факторів (дивергенція обов'язків і ролей при розробці ПЗ, можлива територіальна і культурна віддаленість у складі команди, міні- і мультиекономічні кризи, що почастішали, та ін.) вимагають від розробників гнучкіших підходів і швидкого реагування на зовнішні фактори.

Таблиця 1.1. Результати порівняння звісних методологій розроблення програмного забезпечення

| Методологія розробки ПЗ | Каскадна (Waterfall) | Спіральна (spiral) | Гнучка (Agile) |
|---|---|---|---|
| Тип методології | послідовна (прогнозована) | циклічна (напівпрогнозована) | гнучка (абстрактна) |
| Циклів конструювання | 1 | >1 | >1 |
| Визначення вимог безпеки | + | + | - |
| Прогноз ризиків безпеки | + | + | - |
| Управління ризиками безпеки | - | - | - |
| Підвищена увага питанням тестування безпеки | - | - | - |

У цій ситуації виникає протиріччя між підвищеними вимогами до безпеки ПЗ (врахуванням усіх вразливостей) і необхідністю адаптації до існуючих об'єктивно-суб'єктивних факторів, властивих сучасному світу ІТ-індустрії.

## 1.3 Дослідження та порівняльний аналіз основних підходів математичного моделювання процесу розробки безпечного програмного забезпечення

Аналіз літератури [40-48] показав, що нині існує багато підходів математичного моделювання процесу розроблення програмного забезпечення. Їх основою є положення теорії системного аналізу [49, 50], масового обслуговування [42, 51-54], нейронних мереж [55-57], нечіткої логіки [58-61], комбінаторні методи розрахунку та графові моделі [62-64] та ін. При цьому отримали свій подальший розвиток рішення оптимізаційних задач, що піддалися формалізаці на основі вищерозглянутих моделей, насемперед це евристичні, аналітичні, структурні методи та ін. [65, 67–75]. Розглянемо існуючі рішення та методи розроблення ПЗ з різними рівнями адаптованості до сучасних вимог кібербезпеки. З урахуванням найчастіше використовуваних на практиці моделей розроблення ПЗ при формалізації процесів управління розробкою.

На початку відмітимо, що нині в теорії системного аналізу виділяють ряд напрямів, серед яких виділимо напрями якісного і кількісного аналізу різних технічних систем і процесів. При цьому, якісний аналіз характеризується простотою і високою швидкістю реалізації, а кількісний аналіз точністю.

При якісному аналізі, однією з основних складових є експертна оцінка процесів, що виконується і розраховується з використанням різних методологій.

Окрім цього, необхідно виділити багато методів і інструментів аналізу причинно-наслідкових зв'язків між різними факторами (діаграма Ішикави, методологія ранжирування та ін.).

Для опису поведінки керованих процесів (у тому числі і процесу розроблення ПЗ) з дискретною множиною станів і безперервним часом широко використовується теорія марківських процесів.

Якщо при цьому закони розподілу тривалості перебування в кожному зі станів до відходу в інший можливий стан не є експоненційними, то адекватною моделлю поведінки системи є напівмарківський процес [76-78].

Класичні розрахунки фінального розподілу ймовірностей відносно станів системи на основі напівмарківських систем проходять за формулами:

$$P_i = \pi_j \overline{T}_j / \sum_{j=1}^{n} \pi_j \overline{T}_j \ , \ i = 1,2,..,n$$

де $n$ – кількісні значення можливих станів системи, $\overline{T}_j$ – перебування у стані *i* до відходу (усереднений час), $\pi_i$ –ймовірність перебування у стані *i*.

$$\overline{T}_j = \int_0^{\infty} (1 - F_i(t))dt = \int_0^{\infty} (1 - \sum_{j=1}^{n} P_{ij} F_{ij}(t))dt$$

Якщо окрім фінальних ймовірностей для дослідження системи необхідне знання яких-небудь часових характеристик її поведінки (наприклад, закон розподілу часового інтервалу), доцільно використати апарат інтервально-перехідних ймовірностей [79].

З літератури відомо що закон розподілу часу перебування в кожному зі станів системи не є обов’язково експоненційним, а може бути довільним це характеризує та відрізняє напівмарківський процес від марківського.

Ці фактори доцільно використовувати при кількісній оцінці вразливостей розроблення ПЗ для розроблення оптимізаційної стратегії прийняття рішень.

Одним із сучасних напрямів математичного моделювання є біологічний напрям за допомогою нейронних мереж [80]. Багато в чому це пов’язано із специфікою функціонування комп’ютерних систем, які є людино-машинними системами.

Крім того, останнім часом все більше уваги розробники і проектувальники почали приділяти питанням захисту даних від програмних загроз. А в цьому випадку результати дослідження систем, проведені за допомогою біологічного підходу, показують найбільш адекватні результати [81-90].

Проведені дослідження моделей КС, представлених у вигляді нейронних мереж [55, 91-93], показали що разом з їх перевагами існують значні недоліки які пов'язані з істотними часовими витратами на систематичний процес навчання при побудові необхідної моделі. Що в наслідку приводить до реакційності по відношенню до швидких динамічних змін в процесі управління розробленням ПЗ.

Тому ці моделі доцільно використовувати при моделюванні окремих компонентів або структурних елементів інтелектуальних систем прийняття рішень або використовувати в основі процесу формування практичних рекомендацій менеджерам [94-101].

Однією з поширених технологій математичної формалізації процесів, що протікають в технічних системах, є технологія автоматного моделювання. В автоматній моделі управління технологія розроблення безпечного ПЗ представляється детермінованим автоматом, на вхід якого поступає послідовність команд користувачів.

Основними елементами автоматної моделі може бути: множина станів системи $\{V\}$, множина користувачів $\{U\}$, множина матриць доступів $\{M\}$, багато команд користувачів, що змінюють матрицю доступів $\{CC\}$, багато команд користувачів, що змінюють стан $\{VC\}$, багато вихідних значень $\{Out\}$ [79, 102-103].

Перевагою цієї технології є можливість відображення різних підходів управління, що визначають не лише архітектуру системи, але і конфігурацію, порядок взаємодії між об'єктами і суб'єктами процесу розроблення ПЗ.

Серед недоліків автоматних моделей можна відмітити складність їх практичної реалізації у разі збільшення використовуваних підходів і методологій розроблення безпечного ПЗ.

Як відмічено у ряді джерел [51, 63, 104, 105], графокомбінаторний підхід на протязі значного часу був пріоритетним для розвязання задач аналізу та синтезу систем управління різного призначення

Використовуючи вищезазначений підхід можна представити процес розробки безпечного ПЗ у вигляді функції: $G(N,C)$ або $G(\text{x})$, де $N$ – множина можливих станів під час управління, $C$- множина можливих зв'язків між цими станами, $x$ – одна з характеристик якості управління: безпека, вартість, ефективність та ін., що визначається як критерій оптимізації.

Тоді окрема задача розроблення безпечного ПЗ може трансформуватися в оптимізаційну в наступну задачу оптимізації:

$$G(x) \to opt .$$

Вирішення мережевих задач управління у рамках цього підходу спрямоване на розгляд можливих варіантів для досягнення оптимуму відносно властивості, що аналізується, ґрунтуючись на моделюванні процесу у вигляді графа.

Одним з можливих методів математичного графового представлення процесу розроблення безпечного ПЗ є метод, заснований на *GERT*-мережах. Цей метод успішно застосований при формалізації деяких процесів, що пов'язані з проектуванням і тестуванням ПЗ в зазначених роботах [53, 62, 106, 107] та дає змогу, навіть з невідомою функцією розподілу випадкових величин заздалегідь, моделювати процеси.

Отримані результати моделювання зазначених процесів підтвердили, що враховуючи ймовірні негативні наслідки і ситуації можна застосовувати цей підхід.

То ж доречно використовувати еквівалентні спрощення складних процесів, розподіл на підетапи складних етапів, відомі математичні апарати, такі як: леми Жордана, формули Мейсона та ін. згідно аналізу *GERT*-моделей.

На рис. 1.4 представлений порівняльний аналіз існуючих моделей управління розробленням безпечного ПЗ відзначивши переваги та недоліки.

| **МОДЕЛІ УПРАВЛІННЯ РОЗРОБЛЕННЯМ БЕЗПЕЧНОГО ПЗ** | | | |
|---|---|---|---|
| **Нейронні мережі** | **Апарат випадкових процесів** | **Автоматні** | **Графові** |
| **Переваги:**<br>– можливість моделювання адаптивних систем;<br>– можливість врахування фактору апріорної невизначеності вхідних сигналів;<br>– можливість урахування специфіки зовнішніх впливів. | **Переваги:**<br>– можливість реалізації в системах у вигляді монітора посилань і системи аудиту. | **Переваги:**<br>– різноманіття політик управління, що визначають порядок взаємодії суб'єктів і об'єктів управління між собою. | **Переваги:**<br>– можливість визначення довільних функцій розподілу випадкових величин;<br>– простота реалізації. |
| **Недоліки:**<br>– необхідність розбиття моделі на ряд простих моделей;<br>– складність опису аналітичного представлення. | **Недоліки:**<br>– відсутність урахування специфіки методології SCRUM і вимог безпеки. | **Недоліки:**<br>– складність практичної реалізації. | **Недоліки:**<br>– не враховуються змінювані і підлаштовувані в процесі функціонування параметри. |
| **Загальний недолік:** Складність врахування фактору апріорної невизначеності в параметрах безпеки, відсутність врахування в моделях динамічних змін. | | | |

Рис. 1.4. Порівняльний аналіз існуючих моделей управління розробленням безпечного ПЗ

Таким чином, в результаті аналізу і порівняльних досліджень існуючих моделей розроблення безпечного ПЗ виявлено деякі характерні особливості, переваги і недоліки вже існуючих напрямів аналізу та об’єднання цих систем.

Більшість моделей, які використовуються при реалізації технології розроблення безпечного ПЗ, не беруть до уваги деякі фактори в параметрах безпеки, а саме апріорної невизначеності, що визначило дослідження основних підходів моделювання.

До того ж необхідні додаткові дослідження і розробки в силу неврахування динамічних змін, що є особливістю *Agile* в моделях при розробці ПЗ.

## 1.4 Постановка завдання дослідження

Проведені дослідження існуючих моделей, методів, методологічного забезпечення процесу розроблення безпечного ПЗ, методів та засобів управління цим процесом, технологій математичної формалізації надав можливість виявити значний ряд недоліків та обмежень на фоні підвищеного інтересу кіберзлочинців до ПЗ та високих вимог безпеки ПЗ.

Ці результати об'єктивно характеризують існуюче протиріччя розвитку методологій розроблення безпечного програмного забезпечення в умовах існуючої філософії *Agile* і гнучких методологій розроблення ПЗ (рис. 1.5.).

Це дозволило зробити висновок про необхідність систематичного врахування комплексу сучасних негативних факторів, що впливають на безпеку інформації, при розроблені, реалізації та впроваджені безпечного ПЗ.

Проведені дослідження дали змогу побачити, що існує широкий діапазон варіантів розроблення і використання сучасних моделей та методів управління розробленням безпечного ПЗ.

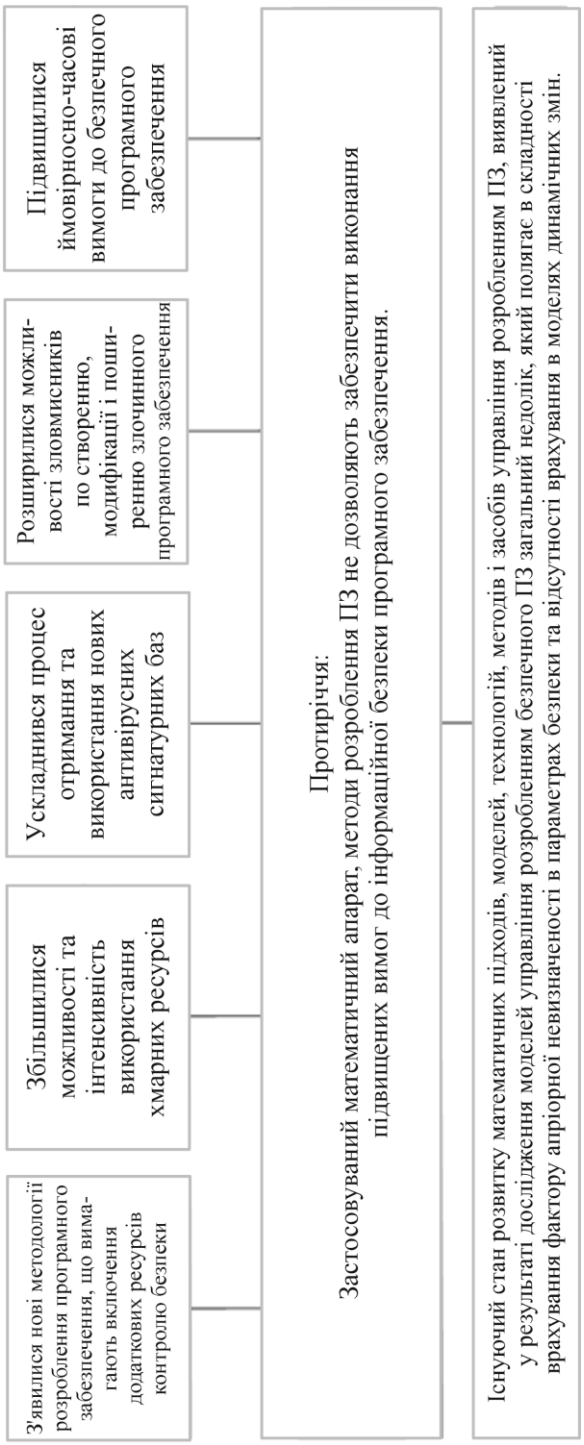


Рис. 1.5. Об'єктивно існуюче протиріччя розвитку моделей та методів розроблення безпечного програмного забезпечення

Вони відрізняються підходами впровадження, властивостями окремих елементів та підсистем, практиками, тактико-технічними показниками, фінансовими характеристиками т.і. [33, 34, 108-112].

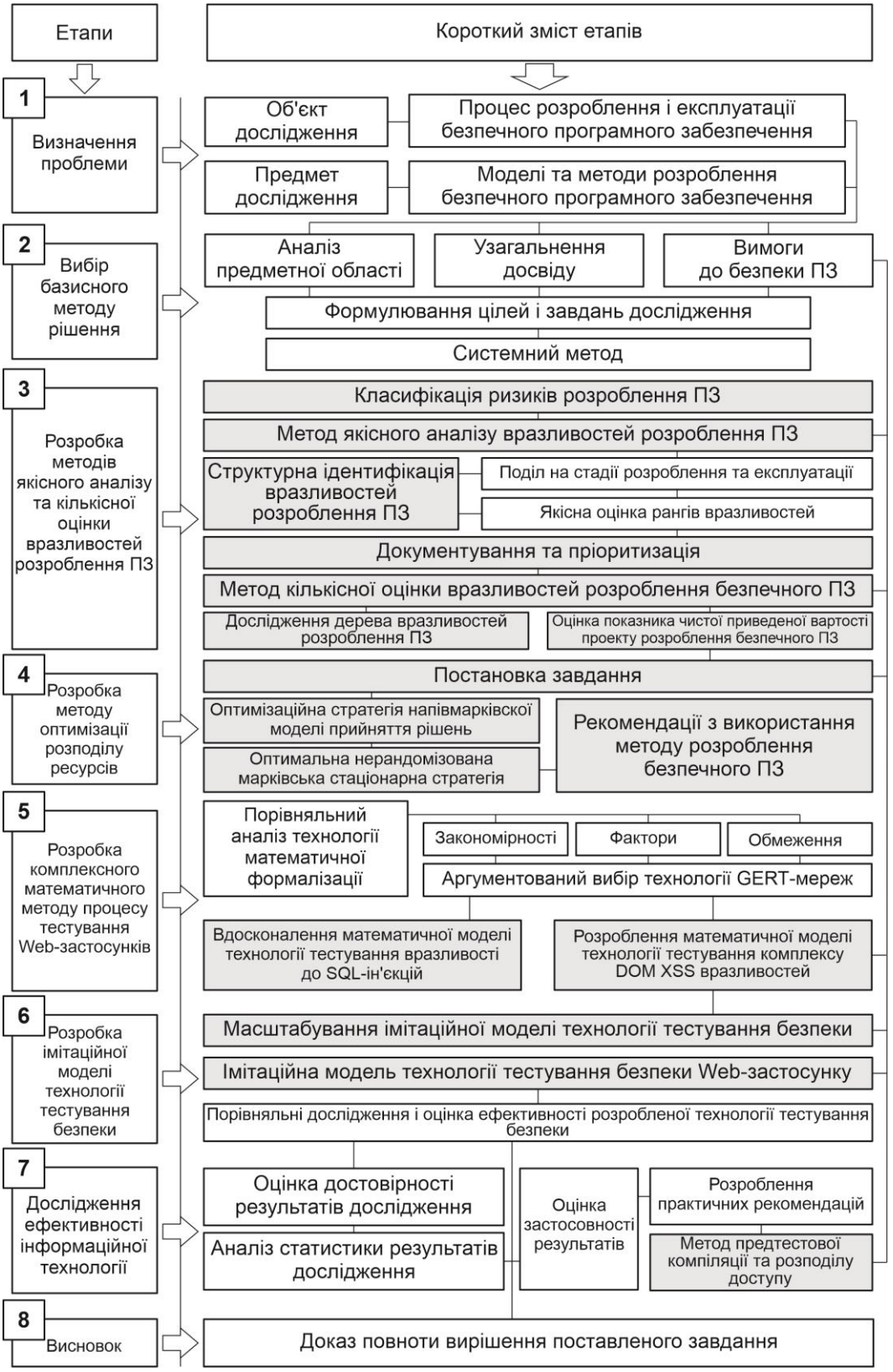


Рис. 1.6. Структурна схема виконання дисертаційного дослідження

Основним завданням для вирішення **науково-технічної проблеми**, яка полягає в синтезі моделей та методів розроблення безпечного ПЗ КС, є

розроблення, вдосконалення і вибір сучасних методів та моделей, що забезпечують максимальні показники безпеки ПЗ.

Вирішення вказаної проблеми включає 8 етапів. Структурна схема дослідження представлена на рис. 1.6.

Для вирішення вказаної науково-технічної проблеми виникає необхідність у вирішенні наступних взаємопов'язаних задач:

– удосконалення методу якісного аналізу вразливостей розроблення програмного забезпечення, відмінною особливістю якого є врахування факторів експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки ПЗ КС і оцінкою довільного несуперечливого кінцевого набору "квантів інформації";

– удосконалення методу кількісної оцінки вразливостей розроблення ПЗ, відмінною особливістю якого є комплексне використання "Аналізу дерева відмов" і способу оцінки показника чистої приведеної вартості проекту розроблення ПЗ з урахуванням негативних факторів можливого невиявлення загроз безпеки ПЗ КС;

– удосконалення методу оптимізації розподілу ресурсів розроблення ПЗ, який відрізняється від відомих використанням псевдобулевих методів бівалентного програмування з нелінійною цільовою функцією і лінійними обмеженнями для визначення оптимальної стратегії усунення експлуатаційних помилок;

– розроблення комплексу математичних моделей технології тестування *Web*-застосунків, в основу яких мають бути покладені принципи і методи *GERT*-моделювання;

– удосконалення імітаційної моделі технології тестування безпеки, що відрізняється від відомих адаптацією вибору вхідних операторів управління і даних до підвищення вимог оперативності розроблення і реалізації моделі, вираженої в реалізації процедури взаємодії із реальним

браузером з використанням засобів автоматизації браузеру і формуванням даних для атаки на декількох діалектах;

– розроблення методу передтестової компіляції і розподілу доступу, відмінною особливістю якого є врахування профілів користувача при синтезі застосунку, а також використання ресурсів "хмарних сховищ" в процесі отримання інсталяційних версій;

– впровадження розроблених моделей, методів та засобів розроблення безпечного ПЗ.

Дисертаційні дослідження повинні здійснюватися на основі систематизації вже відомих підходів в предметних областях програмної і комп'ютерної інженерії та розроблення нових методів їх аналізу та синтезу.

При розробці методу якісного аналізу необхідно відмітити, що основою для розроблення є структурна ідентифікація вразливостей розроблення ПЗ, що відрізняється від відомих побудовою оцінки вразливостей розроблення ПЗ "зверху" у вигляді множини, за наявності довільного несуперечливого кінцевого набору "квантів інформації".

При розробці комплексу математичних моделей технології тестування *Web*-застосунків необхідно виділити наступні окремі наукові завдання: розроблення і дослідження математичних моделей технології тестування *DOM XSS* вразливості і технології тестування вразливості до *SQL*-ін'єкцій, з необхідністю провести оцінку ефективності та адекватності отриманого розробленого комплексу математичних моделей.

## 1.5 Висновки до розділу

Проведено аналіз науково-прикладної проблеми, досліджено сучасний стан та тенденції розвитку моделей та методів розроблення безпечного програмного забезпечення і вимог до програмних засобів, показників і

критеріїв оптимізації, а також підходів математичної формалізації відповідних інформаційних процесів.

Проведені аналіз і дослідження дозволили сформувати ієрархічний показник вимог до якості ПЗ КС, і виділити в ньому вимоги до безпеки.

У результаті сформована загальна схема характеристик і показників, що відносяться до якості програмного забезпечення.

Аналіз моделей та методів і методологій розроблення безпечного програмного забезпечення, а також факторів, що впливають на безпеку, дозволив виділити протиріччя між підвищеними вимогами до безпеки ПЗ (врахуванням усіх вразливостей) і необхідністю адаптації до існуючих об'єктивно-суб'єктивних факторів, властивих сучасному світу IT-індустрії.

Проведені порівняльні дослідження основних підходів математичної формалізації методологій розроблення безпечного ПЗ дозволили визначити основні напрями дисертаційного дослідження і сформулювати оптимізаційне завдання синтезу моделей та методів розроблення ПЗ для підвищення безпеки даних.

Сформульовано завдання дисертаційного дослідження, в якому визначено, що основним завданням синтезу моделей та методів розроблення ПЗ є забезпечення максимальних показників безпеки ПЗ.

Представлено розроблену структурну схему проведення дисертаційного дослідження.

Отпубліковані наукові результати роботи автора приведені у 1 розділі [90, 102, 113, 114, 115, 116, 117, 118, 119, 120].

**Список використаних джерел:**


[1] Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Електронний ресурс]. – Режим доступу до ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

[2] Семенов С.Г. Методы и средства распределения доступа и защиты данных в компьютеризированных информационных управляющих системах критического применения / С.Г. Семенов. – Х.:НТУ «ХПІ», 2013. – 360 с.

[3] ДСТУ В 3265–95. Зв'язок військовий. Терміни та визначення. Київ: Український науково-дослідний і навчальний центр проблем стандартизації, 1995. 23с.

[4] ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів. Київ: ДЕРЖСПОЖИВСТАНДАРТ УКРАЇНИ, 2008. 29 с.

[5] ISO/IEC 25012:2008. Software engineering. Software product. Quality Requirements and Evaluation (SQuaRE). Data quality model. 2008. 13 p.

[6] Блинов А.М. Информационная безопасность: учеб. пособие. Часть 1. Санкт-Петербург: Издательство Санкт-петербургского государственного университета экономики и финансов, 2010. 96с.

[7] Надеждин Е.Н. Оценка эффективности механизма защиты сетевых ресурсов на основе игровой модели информационного противоборства. / Е.Н.Надеждин // Научный вестник – Тамбов: ООО "Консалтинговая компания Юком". – 2007. – №2(4) . –С. 49-58

[8] Черушева Т. В. Проектирование программного обеспечения / Т. В. Черушева. – Пенза : Изд-во ПГУ, 2014. – 172 с.

[9] Эффективное использование GNU Make [Електронний ресурс]. – Режим доступу: https://www.opennet.ru/docs/RUS/gnumake/

[10] By Kenneth Secure Coding: Principles and Practices / By Kenneth van Wyk, Mark Graff – Publisher: O'Reilly Media June 2009. – P. 224

[11] IxChariot: Тестирование в условиях атак отказа в обслуживании (DOS) [Електронний ресурс]. – Режим доступу до ресурсу: http://ixchariot.ru/tests/

[12] Sherman M., Schiela R. From Secure Coding to Secure Software / [Електронний ресурс] – Режим доступу: http://www.sei.cmu.edu/webinars/view_webinar.cfm?webinarid=483646

[13] Гавриленко С.Ю. Методика відбору системи показників для ідентифікації стану комп'ютерної системи критичного застосування// Радіоелектронні і комп'ютерні системи/ зб. наук. пр. – Харків: НАУ «ХАІ», 2019. – №2 (90). С.127-135.

[14] Благодатских В.А. Стандартизация разработки программных средств / В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов – М: Финансы и статистика, 2003. – 288 с.

[15] Вендров А.М. Проектирование программного обеспечения экономических информационных систем / А.М. Вендров. – М.: Финансы и статистика, 2005. – 544 с.

[16] Гусятников В.Н. Стандартизация и разработка программных систем / В.Н. Гусятников, А.И. Безруков – М: Финансы и статистика, 2010. – 288 с.

[17] ISO/IEC 27002:2013 Information technology. Security techniques. Code of practice for information security controls. 2013. 80 p.

[18] ISO/IEC TR 18044 Information technology. Security techniques. Information security incident management. 2004. 50 p.

[19] ISO/IEC 27005:2008. Information technology. Security techniques. Information security risk management. 2008. 55 p.

[20] ISO/IEC/IEEE 15939:2017 Systems and software engineering. Measurement process. 2017. 39 p.

[21] ISO/IEC TS 33030:2017 Information technology. Process assessment. An exemplar documented assessment process. 2017. 33 p.

[22] НД ТЗІ 2.5-010-03. Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу. Київ, ДСТСЗІ СБ України. 2012. 20 с.

[23] Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах: Постанова Кабінета Міністрів України від 29.03.2006 №373.

[24] ISO/IEC «Информационная технология Методы и средства обеспечения безопасности - Критерии оценки безопасности ИТ - Часть 1: Введение и общая модель». ISO/IEC JTC 1/SC27 №2738, 02.2001 г.

[25] ISO/IEC 15408 3: 1999 «Информационная технология - Методы и средства обеспечения безопасности - Критерии оценки безопасности ИТ -Часть 3: Гарантийные требования безопасности».

[26] ISO/IEC PDTR 15446 «Информационная технология Методы и средства обеспечения безопасности - Руководство по разработке профилей защиты и заданий по безопасности», ISO/IEC JTC 1/SC27 №2603 dra, 04.2001 г.

[27] Требования к средствам защиты конфиденциальной информации [Електронний ресурс]. – Режим доступу до ресурсу:

http://www.sec4all.net/statea118.html

[28] Denning P.J. Discussing Cyber Attack / P.J. Denning, D.E. Denning // Comm. of the ACM. – 2010. – Vol. 53. – №. 9. – P. 29-31

[29] Gavrylenko S. Software security overview /Anoushirvan Rashidinia, S. Gavrilenko, M. Pochebut, O. Sytnikova// Системи управління навігації та зв'язку. – ПНУ, 2019. – Вип. 2 (54) с.55-58.

[30] Sherman M. Building Secure Software for Mission Critical Systems [Електронний ресурс] – Режим доступу: http://resources.sei.cmu.edu/ asset_files/ Presentation/ 2017_017_001_495865.pdf

[31] Allan, H. Visual Specification of Security / H. Allan, M. W. Maimone, J. D. Tygar, J. M. Wing, M. Zaremski, A. Miry. // IEEE Transactions on Software Engineering (TSE), 1990. – 16(1). – P. 1185-1197.

[32] Вэйдер Майкл Томас. Инструменты бережливого производства. Мини-руководство по внедрению методик бережливого производства / Майкл Томас Вэйдер – Альпина Паблишер. 2012. – 125 с.

[33] Когда «Agile» (не) к месту [Електронний ресурс]. – Режим доступу до ресурсу: https://makhmetov.ru/articles/agile.html

[34] Krishnan M. Soumya Software Development Risk Aspects and Success Frequency on Spiral and Agile Model / M. Soumya Krishnan // International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization). – 2015. –Vol.3. – №1. – pp.301-310

[35] Putu Adi Guna Permana Scrum Method Implementation in a Software Development Project Management / Putu Adi Guna Permana // International Journal of Advanced Computer Science and Applications. – 2015. – Vol. 6. № 9. – P. 199-205.

[36] Бриткин А. И. Риски, связанные с внедрением технологий, в проектах разработки программного обеспечения / А. Бриткин // Социально-экономические и технические системы. – 2007. – № 8 (42).

[37] Hasan Yasar Security Practitioner Perspective on DevOps for Building Secure Solutions / Режим доступу: http://www.sei.cmu.edu/ webinars/view_webinar.cfm?webinarid= 474101&ga Webinar=Security Practitioner PerspectiveonDevOpsforBuildingSecureSolutions

[38] Swiderski Frank Threat Modeling / Frank Swiderski, Window Snyder. – Microsoft Press, 2004. 257 p.

[39] Zhang Peng. Security in Network Coding / Peng Zhang, Chuang Lin. – Springer, 2016. – 98 p.

[40] Квєтний Р. Н. Комп’ютерне моделювання систем та процесів. Методи обчислень. Частина 1 : навчальний посібник / Квєтний Р. Н., Богач І. В., Бойко О. Р., Софина О. Ю., Шушура О.М.; за заг. ред. Р.Н. Квєтного. Вінниця: ВНТУ, 2012. 193 с.

[41] Полицын С. А. Подходы к вычислению временных затрат на проекты в сфере разработки программного обеспечения на основе использования прецедентов. Программная инженерия. 2011. №7. С. 9-14

[42] Семенов С.Г., Лисица Д.А. Модель оценки риска разработки программного обеспечения: сб. материалов XV Международной НТК «Проблемы информатики и моделирования». Харьков: НТУ «ХПИ», 2015. С.82.

[43] Сирота А.А. Компьютерное моделирование и оценка эффективности сложных систем / А.А. Сирота. –М.: Техносфера, 2006. –280 с.

[44] Hybrid A Semi-Supervised Anomaly Detection Model for High-Computational Dimensional Data// [ Hongchao Song, Zhuqing Jiang,

Aidong Men, and Bo Yang ] : Intelligence and Neuroscience. – 2017. – P.9.

[45] Machine Learning Techniques for Anomaly Detection// [ Salima Omar, Asri Ngadi, Hamid H. Jebur ]// International Journal of Computer Applications (0975 – 8887).– 2013, vol.79, No.2.– P. 33-41.

[46] Takagi T. Fuzzy identification of systems and its applications to modeling and control / T. Takagi, M. Sugeno // IEEE Transactions on Systems, Man and Cybernetics. – 1985. – Vol. SMC-15, No. 1. – P. 116–132.

[47] Patcha, A. An overview of anomaly detection techniques: Existing solutions and latest technological trends / A. Patcha, J. M. Park : Computer Networks. – 2007. – Vol. 51, №. 12. – pp. 3448-3470.

[48] Swiderski Frank Threat Modeling / Frank Swiderski, Window Snyder. – Microsoft Press, 2004. 257 p.

[49] Kullback S. Information Theory and Statistics / S. Kullback. – New York: Wiley, 1959. – 395 p.

[50] Ljung L. System identification: theory for the user / L. Ljung. – Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1986. – ISBN 0-138-81640-9.

[51] Семенов С.Г., Подорожняк А.А. , Баленко А.И. Анализ и синтез защищенных компьютерных систем и сетей. Харьков, 2012. 204 с.

[52] Семенов С.Г. Модели и методы управления сетевыми ресурсами в информационно-телекоммуникационных системах: монография / С.Г. Семенов, О.А. Смирнов, Є.В. Мелешко. Х.: НТУ «ХПІ». – 2012. – 212 с.

[53] Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник НТУ «ХПІ». – Х.:НТУ «ХПІ». – 2012.

–№62 (968). – С 173-181.

[54] Rabin, M.O., Probabilistic automata. Information and Control 6(3), 1963. – pp. 230–245.

[55] Hopfield J. J. Neural networks and physical systems with emergent collective computational abilities // Proceedings of National Academy of Sciences. – 1982. – Vol. 79 № 8. – pp. 2554–2558

[56] Kudlek M.. Probability in Petri nets. Fundamenta Informaticae. – 2005 N 67 (1-3). – p.121-130.

[57] Sapozhnikova Elena P.ART-Based Neural Networks for Multi-label classification / E.P. Sapozhnikova // Springer-Verlag Berlin Heidelberg, 2009. – P. 167-177.

[58] Mamdani E. H. Application of fuzzy logic to approximate reasoning using linguistic synthesis / E. H. Mamdani // IEEE Trans. Comput. – 1977. – Vol. 26, No. 12. – P. 1182–1191.

[59] Zadeh L. A. Fuzzy sets / L. A. Zadeh // Information and Control. – 1965. – Vol. 8, No. 3. – P. 338–353.

[60] Novák Vilém Mathematical Principles of Fuzzy Logic / Vilém Novák, Irina Perfilieva, J. Mockor. 2012 – 320 p

[61] Sarlin. P. Fuzzy clustering of the self-organizing map: some applications on financial time series / P. Sarlin, T Eklund : Advances in self-organizing maps. – Springer Berlin Heidelberg, 2011. – P. 40-50.

[62] Халифе К. GERT-модель процесса безопасного тестирования программного обеспечения / Кассем Халіфе, А.Е. Горюшкіна, В.М. Зміївська // Зб. наукових праць. Системи обробки інформації. – Х.: ХУ ПС, 2016. – Випуск 3(140). – С.21-24.

[63] Koller D. and N.Friedman, Probabilistic Graphical Models. Principles and Techniques. Massachusetts: MIT Press, 2009. – 1280 p.

[64] Kind A., Histogram-based traffic anomaly detection / A. Kind, M. P. Stoecklin and X. Dimitropoulos : IEEE Transactions on Network and Service Management, vol. 6(2), 2009. cP. 110-121.

[65] Arthur Zimek There and back again: Outlier detection between statistical reasoning and data mining algorithms /Arthur Zimek, Peter Filzmoser // Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. – 2018, Т. 8, вып. 6. – P.22-29.

[66] Jonsson B., Larsen K.G., and Yi W. Probabilistic extensions of process algebras, Handbook of Process Algebras, Elsevier, North Holland, – 2001.

[67] Raskin, L. Method of solving fuzzy problems of mathematical programming / L. Raskin, O. Sira // Eastern-European Journal of Enterprise Technologies. – 2016. – Vol. 5, No. 4 (83). – P. 23–28.

[68] Amitava Mitra. An adaptive exponentially weighted moving average-type control chart to monitor the process mean. European Journal of Operational Research. Volume 279, Issue 3, 16 December 2019,– P. 902-911.

[69] Liu Y., Miao H., Zeng H., and Li Z. Probabilistic Petri net and its logical semantics. In Software Engineering Research, Management and Applications, 2011. – p. 73–78.

[70] Markus Goldstein. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data./ Markus Goldstein andSeiichi Uchida// PLoS ONE. – 2016, vol. 11, no. 4, P.28-32.

[71] Mateus, P., Qiu, D., Li, L. On the complexity of minimizing probabilistic and quantum automata. Information and Computation.– 2012.– 36–53.

[72] Patcha, A. An overview of anomaly detection techniques: Existing solutions and latest technological trends / A. Patcha, J. M. Park : Computer Networks. – 2007. – Vol. 51, №. 12. – pp. 3448-3470.

[73] Rabin, M.O., Probabilistic automata. Information and Control 6(3),

1963. – pp. 230–245.

[74] Stoelinga M. An introduction to probabilistic automata, Bulletin of the European Association for Theoretical Computer Science, 2002., – p. 176–198.

[75] Гусятников В.Н. Стандартизация и разработка программных систем / В.Н. Гусятников, А.И. Безруков – М: Финансы и статистика, 2010. – 288 с.

[76] Markou, M. Novelty detection: a review – part 1: statistical approaches / M. Markou, S. Singh // Signal processing. – 2003. – Vol. 83, №. 12. – pp. 2481-2497.

[77] Markou, M. Novelty detection: a review – part 2: neural network based approaches / M. Markou, S. Singh //Signal processing. – 2003. – Vol. 83, №. 12. – pp. 2499-2521.

[78] Smyth, P. Clustering sequences with hidden Markov models / P. Smyth : Advances in neural information processing systems. – 1997. – P. 648-654.

[79] Gavrylenko S. Development of anomalous computer behavior detection method based on probabilistic automaton. Monograph/ Gavrylenko S., Chelak V., Semenov S., E. Chelak/ Kiev, 2019. – c. 237-258

[80] Gavrylenko S. Development of the disable software reporting system on the basis of the neural network/ S. Gavrylenko, O. Babenko, E. Ignatova// Journal of physics: Series 998(2018)012009, 2018. – P. 1-8.

[81] Кузнецов О.О. Протоколи захисту інформації у комп'ютерних системах та мережах / О.О. Кузнецов, С.Г. Семенов. – Х.: ХНУРЕ, 2009. – 184 с.

[82] Лужецький В.А. Організаційно-правові питання безпеки інформації Концептуальна модель системи інформаційного впливу / В.А. Лужецький // Безпека інформації. Ukrainian Scientific Journal of

Information Security Том 23, № 1 (2017).

[83] Hasan Yasar Security Practitioner Perspective on DevOps for Building Secure Solutions / Режим доступу: http://www.sei.cmu.edu/ webinars/view_webinar.cfm?webinarid= 474101&gaWebinar= Security PractitionerPerspectiveonDevOpsforBuildingSecureSolutions

[84] OSSTMM 3. Open-Source Security Testing Methodology Manual [Електронний ресурс] – Режим доступу: http://www.isecom.org/ mirror/OSSTMM.3.pdf

[85] Parker, C. B. and E. R. DeLong .ROC methodology within a monitoring framework. Stat Med 22(22): 2003.– c.347-348.

[86] Portnoy L. “Intrusion Detection with Unlabeled Data Using Clustering” / L. Portnoy, E. Eskin, S. J. Stolfo, Columbia University, New York, 2001.

[87] Seacord Robert Secure Coding in C and C++ / Robert C. Seacord – Addison-Wesley, 2013. – 600 p.

[88] Seacord Robert The CERT® C Coding Standard, Second Edition: 98 Rules for Developing Safe, Reliable, and Secure / Robert C. Seacord – Addison-Wesley, 2014. - pp 576.

[89] Seacord Robert The CERT C Secure Coding Standard / Robert C. Seacord – Addison-Wesley, 2008. - pp 720.

[90] Коваленко О.В. Аналіз основних підходів математичного моделювання та методологій для забезпечення максимальних показників безпеки програмного забезпечення / О.В. Коваленко, А.С. Коваленко // Збірник наукових праць всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених «Комп’ютерна інженерія і кібербезпека : досягнення та інновації, м. Кропивницькй. 27-29 листопада 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 74.

[91] Коваленко А.В. Комплекс математических моделей технологии

тестирования web-приложений / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

[92] Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

[93] Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

[94] Мораско Дж. IT-проекты: фронтовые очерки. СПб.: Символ-Плюс, 2007. 384 с.

[95] Макконнелл С. Совершенный код. Мастер-класс. М.: Рус. Редакция ; СПб.: Питер, 2007. 896 с.

[96] Combee Janine Discussion paper Soft controls What are the starting points for the internal auditor? / Janine Combee, Mandy Dijkman, The institute of Internal auditors Niderlands, 2015 [Електронний ресурс] – Режим доступу: https://www.iia.nl/SiteFiles/Publicaties/IIA_Bro%20A4%20Soft%20Controls%20Engels%2002.pdf

[97] Klieber W. Automated Code Repair Based on Inferred Specifications / W. Klieber, W. Snavely – [Електронний ресурс] – Режим доступу: http://resources.sei.cmu.edu/asset_files/ConferencePaper /2016_021_001_483599.pdf

[98] Pedregosa F. et al. Scikit-learn: Machine learning in Python // Journal of Machine Learning Research. – 2011. – P. 2825-2830.

[99] Seacord Robert Modernizing Legacy Systems / Robert C. Seacord,

Daniel Plakosh, Grace A. Lewis – Addison-Wesley, 2003. –332 p.

[100] Seacord Robert Secure Coding in C and C++ / Robert C. Seacord – Addison-Wesley, 2005. –358 p.

[101] Коваленко А.В. Исследование разработанной методики структурной идентификации рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез VIII міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 28-29 квітня 2016 р. – Харків: ХНЕУ. – 2016. – С. 49.

[102] Коваленко О.В. Моделі та методи розробки програмного забезпечення комп’ютерних систем для підвищення безпеки даних: монография / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 350 с.

[103] Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов, // Збірник тез X міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

[104] Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты / В.В. Домарев. – К.: ООО "ТИД "ДС", 2002. – 688 с.

[105] Корченко А. О. Методи ідентифікації аномальних станів для систем виявлення вторгнень. Дис.док.тех.н Спеціальність 05.13.21 – «Системи захисту інформації».–К: 2019.– 405 с.

[106] Шибанов, А.П. Обобщенные GERT-сети для моделирования протоколов, алгоритмов и программ телекоммуникационных систем: дис. доктора техн. наук: 05.13.13 [Текст] / Шибанов Александр

Петрович. – Рязань, 2003. – 307 с.

[107] Pritsker, A. B. GERT: Graphical Evaluation and ReviewTechnique. Part I. Fundamentals / A. B. Pritsker, W. W. Happ The Journal of Industrial Engineering. – 1966. – №5. – P. 681-689.

[108] Ruby Sam Agile Web Development with Rails / Sam Ruby, Dave Thomas, David Heinemeier Hansson – The Pragmatic Bookshelf, 2011. – 480 с.

[109] Lin W-C. An intrusion detection system based on combining cluster centers and nearest neighbors/ W-C. Lin, S-W. Ke, C-F. Tsai : Knowledge-Based Systems.– 2015. – vol. 78.– pp. 13-21.

[110] Hybrid A Semi-Supervised Anomaly Detection Model for High-Computational Dimensional Data// [ Hongchao Song, Zhuqing Jiang, Aidong Men, and Bo Yang ] : Intelligence and Neuroscience. – 2017. – P.9.

[111] Технология разработки программного обеспечения: учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул; под. ред. Л.Г. Гагариной. Москва, ИД «ФОРУМ», 2018. 400 с.

[112] Khalife Kassem Development of Gert model of management system by using test cases / Kassem Khalife, S. G. Semenov, V N. Zmiyevskaya // Journal of Qafqaz university-mathematics and computer science. – 2016. – Vol.(4). – №1 P. 52-59

[113] Коваленко А.В. Задачи распознавания ситуаций в ERP системах/ А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

[114] Коваленко О.В. Управління ризиками розробки програмного забезпечення за умови обмеженості коштів виділених на усунення

помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький: ЦНТУ. - 2018. – С. 128-140

[115] Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44

[116] Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.А. Смирнов, А.В. Коваленко // Збірник наукових праць II міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 24-27 лютого 2016 р. – Київ: Європейський університет. – 2016. – С. 138-139.

[117] Коваленко А.В. Анализ и оценка рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез «Securitea informationala 2015-2016». Conferenta internationala (editia a XII-a). Chisinau. Moldova. 3 martie 2016. – Chisinau: ADSEM. – 2016. – P. 96-102.

[118] Коваленко А.В. Исследование источников и причин риска разработки программного обеспечения, этапов и работ, при выполнении которых возникает риск / А.А. Смирнов, А.В. Коваленко // Збірник тез VII всеукраїнської науково-практичної конференції "Інформатика та системні науки (ІСН-2016)". м. Полтава. 10-12 березня 2016 р. – Полтава.: ПУЕТ – 2016. – С. 264-266.

[119] Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T.

«Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri? eid=2-s2.0-8508321 4331 &origin= AuthorNamesList& txGid= 5633f ba89 7776 a6e0f 3d5633fbcd3fbe **(Scopus)**

[120] Коваленко О.В. Аналіз та дослідження інформаційних технологій розробки програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

# РОЗДІЛ 2
# СИНТЕЗ МЕТОДІВ ЯКІСНОГО АНАЛІЗУ ТА КІЛЬКІСНОЇ ОЦІНКИ ВРАЗЛИВОСТЕЙ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі удосконалено методи якісного аналізу і кількісної оцінки вразливостей розроблення програмного забезпечення, що дозволило вирішити протиріччя, що виникають при розробці ПЗ, і які полягають у зневазі фірмами-розробниками ПЗ факторами вразливості безпеки ПЗ. В якості вирішення вказаної проблеми запропоновано використання розроблених методів якісного аналізу і кількісної оцінки вразливостей розроблення програмного забезпечення.

Удосконалено метод якісного аналізу вразливостей розроблення програмного забезпечення. Його відмінною особливістю є врахування факторів експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки ПЗ і оцінкою довільного несуперечливого кінцевого набору "квантів інформації".

Удосконалено метод кількісної оцінки вразливостей розроблення ПЗ. Його відмінною особливістю є комплексне використання "Аналізу дерева відмов" і способу оцінки показника чистої приведеної вартості проекту розроблення ПЗ з урахуванням негативних факторів можливого невиявлення загроз безпеки ПЗ.

## 2.1 Проблеми аналізу та оцінки вразливостей інформаційної діяльності

Загальні процеси глобалізації економічних, фінансових, соціальних та інформаційних взаємовідношень сприяли розвитку напряму ризик-менеджменту. Проте загальносвітові фінансові кризи показали недостатньо

уважне відношення до управління вразливостями з боку більшості представників керівництва організацій, у тому числі і в Україні.

На теперішній час у більшості організацій і підприємств різних форм власності все більше уваги приділяється питанням аналізу й оцінки вразливостей. Але, незважаючи на це, проблеми і питання, що відносяться до загальної теорії та методології аналізу, оцінки і управління вразливостями вимагають адаптації до підходів і положень сучасного менеджменту, врахуванні нових факторів становлення і розвитку технологій, об'єднання відомих "сталих" положень теорії вразливостей з новими прогресуючими підходами аналізу та синтезу.

Аналіз літератури [1-9, 12, 17] показав, що незважаючи на досить глибоку історію розвитку поняття "вразливість" і спроби ряду відомих авторів сконцентрувати свої розроблення в області управління вразливостями окремих галузей і напрямів діяльності, розроблення нових перспективних наукових положень в цій області все ж дещо "завужена" фінансовою діяльністю. В той же час, широке використання у даній роботі моделей та методів вимагає підвищеної уваги до цього напряму, і відповідно, глибшого освітлення питань ризик-менеджменту IT-індустрії.

Суттю будь-якого процесу, явища або об'єкту (у тому числі і інформаційної складової) є діяльність, яка призводить до формування результатів. У додатку до такого напряму діяльності як розроблення програмного забезпечення, кінцевим результатом, у більшості практичних випадків, є виконання вимог замовника і впровадження розробленого продукту. У сучасних авторів [9, 17-24] дуже часто результат оцінки вразливостей означає можливість або ймовірність настання подій з конкретними наслідками в результаті певних рішень або дій.

Доцільність такого представлення понять в теорії вразливості особливо підкреслюється закономірностями, що виникають в інформаційних

відносинах при розробці безпечного програмного забезпечення, де складність і динаміка взаємозв'язків, нечіткість зовнішніх факторів, а також гетерогенність в структурній і функціональній побудові систем дозволяє розширити класифікацію результатів інформаційної діяльності до виду, представленого на рис. 2.1.


Рис. 2.1. Класифікація результатів інформаційної діяльності

Слід зауважити, що об'єктивний результат є наслідком цілеспрямованого і явного виконання процесу, який пов'язаний з його суттю. Суб'єктивні результати проявляються в тих випадках, коли виконання процесу проходить з недостатнім рівнем визначеності і повноти інформації. На практиці, у сфері IT-індустрії переважаюча кількість вразливостей пов'язана саме з суб'єктивними результатами здійснення ходу або виконання процесу.

Отримання необхідної інформації пов'язане з наявністю чітких і визначених (стандартизованих, апробованих, регламентованих і т.і.) засобів, інструментів, методів та методик, виконання яких пов'язане з ресурсними витратами, а також відсутністю достовірних даних про мету і суть досліджуваного процесу.

Таким чином, можна відмітити, що всі вразливості при розробці програмного забезпечення, з більшим або меншим припущенням, можна вважати суб'єктивним результатом виконання процесу, який пов'язаний з нестачею кількісної або якісної інформації про процес, а також її невизначеністю. Вказані фактори можна вважати головною причиною, яка породжує і супроводжує вразливості в усьому їх життєвому циклі.

Кожну вразливость циклу розроблення програмного забезпечення можна зв'язати з одним із наступних компонентів: дані; людина; система. При цьому слід врахувати міру впливу і відповідальності результатів оцінки вразливостей для різних методологій розроблення безпечного програмного забезпечення.

Аналіз літератури [9, 13-16, 25-27] показав, що в теперішній час існує багато різноманітних методик розроблення безпечного ПЗ. Слід зауважити, що вибір безпосередньо методики при реалізації проекту має суттєвий вплив на результати аналізу, оцінки і управління вразливостями. Наприклад, з літератури [28] відомо, що однією з широко використовуваних методологій розроблення ПЗ є спіральна методологія. Запропонована в 1988 році американським фахівцем Баррі Бом (*Barry Boehm*) [9, 29], ця методологія керується інкрементними розробленнями на основі вразливостей (рис. 2.2).

Як видно з рис. 2.2, більше 15% часових витрат управління IT-проектами йде на аналіз і оцінку вразливостей. При цьому слід зауважити, що на кожному витку "спіралі" це завдання має свої особливості і обмеження, що мають вплив на процес управління вразливостями в системі.

Аналіз літератури [12, 28] показав, що сучасні автори у своїй більшості виділяють п'ять основних вразливостей: помилки, властиві розкладу; поява нових вимог; зміна співробітників; декомпозиція специфікації; низька продуктивність.

Проведені дослідження показали, що ця позиція суперечлива, оскільки не враховує ряд важливих аспектів розроблення безпечного ПЗ. Аналіз нормативної документації ряду відомих фірм-розробників ПЗ показав, що на етапі оцінки вразливостей, як правило, не враховуються вразливості, пов'язані з можливою наявністю помилок в моделях, алгоритмах, програмах обробки інформації, які використовуються для вироблення керуючих рішень, нехтуються вразливості безпеки (можливих помилок, що впливають на вразливість ПЗ).

Це найчастіше призводить до помилок і, відповідно, до необґрунтованих втрат (часових, економічних, іміджевих та ін.).

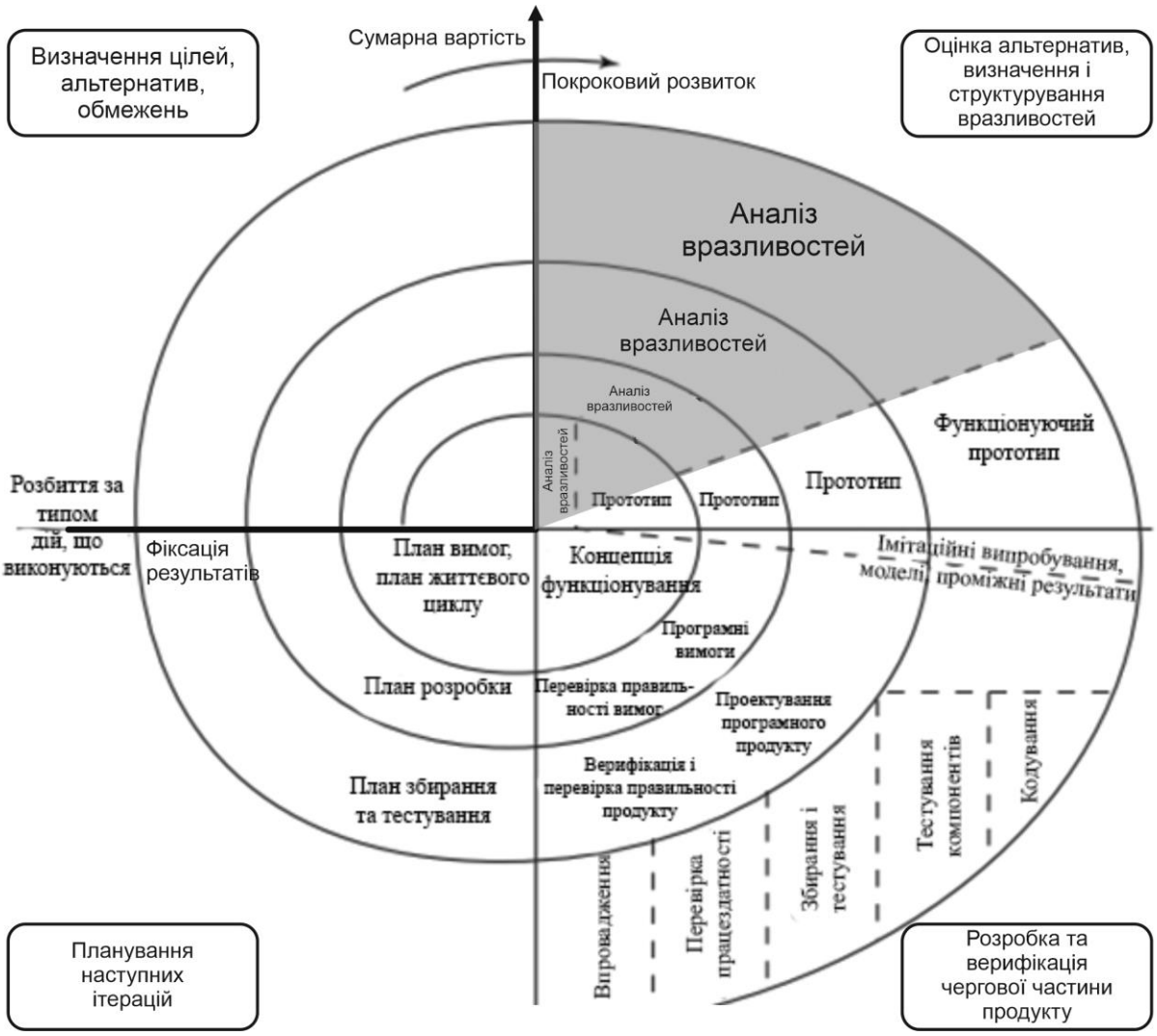


Рис. 2.2. Спіральна модель розроблення ПЗ

Таким чином, проведені дослідження показали, що, незважаючи на важливість рішення задачі управління вразливостями при розробці ПЗ, на даний момент немає чітко сформованої, стандартизованої методологічної бази опису цього процесу. На теперішній час спостерігається:

– відсутність єдиного комплексного і системного погляду на проблему виникнення вразливостей при розробці ПЗ;

– відсутність ясності та прозорості у розумінні кінцевих результатів впливу вразливостей, їх недостатнього враховування при розробці ПЗ;

– значні різночитання в розумінні методик аналізу, оцінки і управління вразливостями;

– недостатність врахування важливих факторів, що виникають у міру вдосконалення технологій і засобів розроблення ПЗ.

Аналіз літератури [8, 17] і проведені дослідження показали, що загальна послідовність оцінки вразливостей найчастіше включає в себе наступні дії:

1. Виявлення джерел і причин вразливості розроблення ПЗ, етапів і робіт, при виконанні яких виникає небезпека.
2. Ідентифікація усіх можливих вразливостей, властивих проекту, що розглядається.
3. Документування результатів та їх наступна пріоритизація.
4. Оцінка рівня окремих вразливостей і вразливості проекта в цілому, яка визначає його економічну доцільність.
5. Визначення допустимого рівня вразливості розроблення ПЗ.
6. Розроблення заходів зі зниження вразливості.

Відповідно до цього алгоритму, оцінка вразливостей розподіляється на три взаємодоповнюючих напрями: якісний (етапи 1, 2, 3) і кількісний аналіз (етапи 4, 5) вразливостей розроблення ПЗ, а також управління (етап 6).

Дослідимо більш детально метод якісного і кількісного аналізу вразливостей розроблення ПЗ.

## 2.2 Метод якісного аналізу вразливостей розроблення програмного забезпечення

Проведені дослідження показали, що метод якісної оцінки вразливостей проекту є описовою, і являє собою процес, спрямований на виявлення конкретних вразливостей проекту, а також причин, що породжують їх, з подальшою оцінкою можливих наслідків і вироблення заходів по роботі з вразливостями. В процесі якісного аналізу вразливостей відбувається вироблення метрик, що відповідають за визначення граничних показників факторів, символізуючих прояви вразливості.

### 2.2.1 Виявлення джерел і причин вразливостей розроблення ПЗ, етапів і робіт, при виконанні яких виникає вразливість

Розглядаючи перший пункт наведеного вище переліку дій з якісного і кількісного аналізу вразливостей, відмітимо, що початкові дані для виявлення і опису характеристик вразливостей можуть братися з різних джерел, наприклад:

- база знань організації;
- інформація з відкритих джерел, наукових робіт;
- маркетингова аналітика;
- опитування експертів та ін.

Ряд відомих авторів [8-12], провівши дослідження, виявили найбільш поширені вразливості при розробці ПЗ. Наприклад, автори Демарко і Лістер

[31] приводять свій список з п'яти найбільш важливих джерел вразливостей будь-якого проекту розроблення ПЗ:

– недоліки календарного планування;
– плинність кадрів;
– роздування вимог;
– порушення специфікацій;
– низька продуктивність.

Можливо відмітити, що цей перелік має узагальнений характер, що значною мірою ускладнює метричну оцінку наведеного списку.

Баррі Бом в своїй роботі [30] розширює список до 10 найбільш поширених вразливостей програмного проекту:

1. Дефіцит спеціалістів.
2. Нереалістичні терміни та бюджет.
3. Реалізація невідповідної функціональності.
4. Розроблення неправильного інтерфейсу користувача.
5. «Золоте сервування», перфекціонізм, непотрібна оптимізація і відточування деталей.
6. Безперервний потік змін.
7. Нестача інформації про зовнішні компоненти, що визначають оточення системи або залучені в інтеграцію.
8. Недоліки в роботах, що виконуються зовнішніми (по відношенню до проекту) ресурсами.
9. Недостатня продуктивність отримуваної системи.
10. «Розрив» у кваліфікації спеціалістів різних областей знань.

Однак і цей перелік не повний і неструктурований. Це ускладнює процес оцінки взаємовпливу наведених вразливостей один на одного.

Досить детально вразливості були оцінені і класифіковані в роботах [9, 12]. Відповідно до цих досліджень, вразливості класифікуються за наступними ознаками:

– середовище (внутрішній, зовнішній вразливості);

– природа (економічний, технічний, технологічний);

– сфера (вразливість проекту, процесу, продукту);

– рівень (від критичного до незначного вразливості);

– галузь впливу (вразливість невиконання бюджету проекту, вразливість невиконання плану проекту, вразливість невиконання якості проекту);

– ланка управління вразливістю (вразливість окремого процесу, вразливість проекту, вразливість компанії).

Проте подібна класифікація робить акцент на проектах розроблення безпечніх програмних систем, які не пов'язані з процесами їх подальшого впровадження і адаптації систем в умовах конкретної організації, експлуатації в умовах можливих зовнішніх зловмисних впливів. Тому являється доцільною необхідність розглядати окремо:

– організаційні вразливості, які пов'язані з тим, що проект викличе такі зміни в структурі і бізнес-процесах компанії, які нівелюють заплановані вигоди;

– операційні вразливості, пов'язані з неконтрольованим зростанням витрат на експлуатацію системи;

– соціальні вразливості, пов'язані з неадекватною поведінкою учасників проекту;

– експлуатаційні вразливості, пов'язані з можливими майбутніми фінансовими, іміджевими і іншими втратами у разі наявності потенційних вразливостей проекту.

### 2.2.2 Структурна ідентифікація вразливостей розроблення програмного забезпечення

Використовуючи результати досліджень наведених вище авторів [12, 17, 30, 31], думки експертів, маркетингові дані, а також бази знань таких відомих фірм, як *Epam Systems* та *Nix Solutions Ltd*, ідентифікуємо вразливості розроблення ПЗ і представимо результат у вигляді структурної схеми класифікації рис. 2.3.

Як видно з рис. 2.3, основні вразливості розроблення програмного забезпечення можна представити у вигляді сукупності множин організаційних $Z = \{Id\,1, \ldots, Id\,5\}$, управлінських $U = \{Id\,6, \ldots, Id\,9\}$, операційних $Y = \{Id\,10, \ldots, Id\,15\}$, технологічних $T = \{Id\,16, \ldots, Id\,20\}$, експлуатаційних $E = \{Id\,21, \ldots, Id\,24\}$, соціальних $C = \{Id\,25, \ldots, Id\,27\}$ та правових $W = \{Id\,28, Id\,29\}$ вразливостей.

Відмінною особливістю представленої класифікації є врахування експлуатаційних вразливостей.

Особливої важливості ці вразливості набувають в умовах підвищеного рівня кіберзлочинності, коли зневага вразливостями програмного забезпечення може призвести до експлуатаційних проблем, а часто і неможливості експлуатації ("краху") ПЗ.

Окрім цього, в умовах українського правового поля спостерігаються окремі випадки неадекватності і невідповідності правовим нормам дій посадовців державного апарату.

Практика ряду відомих фірм-розробників ПЗ (*Nix Solutions Ltd.* та ін.) показує, що вказаний фактор вразливості доцільно враховувати при розробці безпечного ПЗ, разом з фактором можливої зміни українського законодавства.

Вплив вказаних на рис. 2.3 вразливостей на основні фактори успіху розроблення, впровадження і тривалої експлуатації безпечного ПЗ проілюстровано на рис. 2.4.

Як видно з цього рисунка, більшість із розглянутих вразливостей розроблення ПЗ (організаційні, операційні, управлінські та ін.) можуть мати безпосередній вплив як на процес розроблення ПЗ, так і на процес його експлуатації.

В той же час, наприклад, експлуатаційні риски безпосереднього впливу на процес розроблення ПЗ не мають. Але зневага цими вразливостями веде найчастіше до провалу експлуатації ПЗ і втрат майбутніх замовлень і проектів (простоюванням розробників ПЗ). Саме цим фактором викликаний зв'язок між блоками "Провал при експлуатації ПЗ" і "Провал при розробці ПЗ".

Однак, незважаючи на це, в цілому можна виділити множину вразливостей, що безпосередньо впливають на процес розроблення ПЗ:

$$MR = \{Z, U, Y, C, T, W\}$$

і множину вразливостей, що безпосередньо впливають на процес експлуатації ПЗ:

$$ME = \{Z, U, Y, C, T, W, E\}, (Id\ 9, Id\ 10, Id\ 15, Id\ 29 \notin ME).$$

Слід зауважити, що виділені на рис. 2.4. фактори достатньою мірою описують перелік можливих вразливостей розроблення ПЗ. Проте, вони не дають уявлення про взаємний вплив і відповідно можливу зміну кінцевого результату.

Окрім цього, наведені множини вразливостей розроблення ПЗ різною мірою впливають на кінцевий результат.

Тому наступним кроком ідентифікації вразливостей розроблення ПЗ доцільно виконати процедури ранжирування і виділення найбільш пріоритетних (важливих) вразливостей розроблення ПЗ.

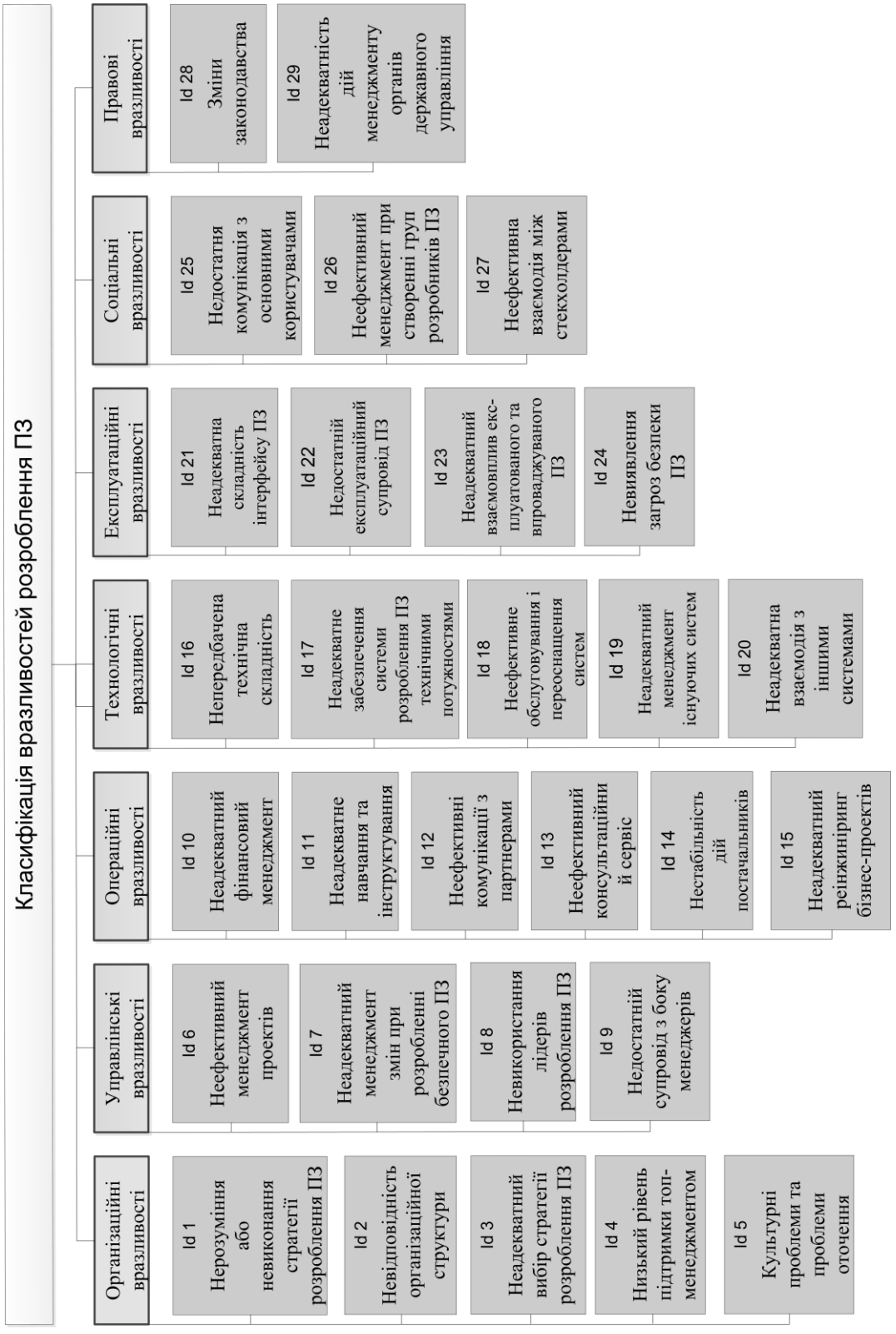


Рис. 2.3. Класифікація вразливостей розроблення ПЗ

Проведені дослідження показали, що для вирішення задачі визначення взаємовпливу вразливостей доцільно використовувати інструмент аналізу

причинно-наслідкових зв'язків між різними факторами і вразливостями, розроблений Каору Ішикава [32, 33] (діаграма Ішикави).

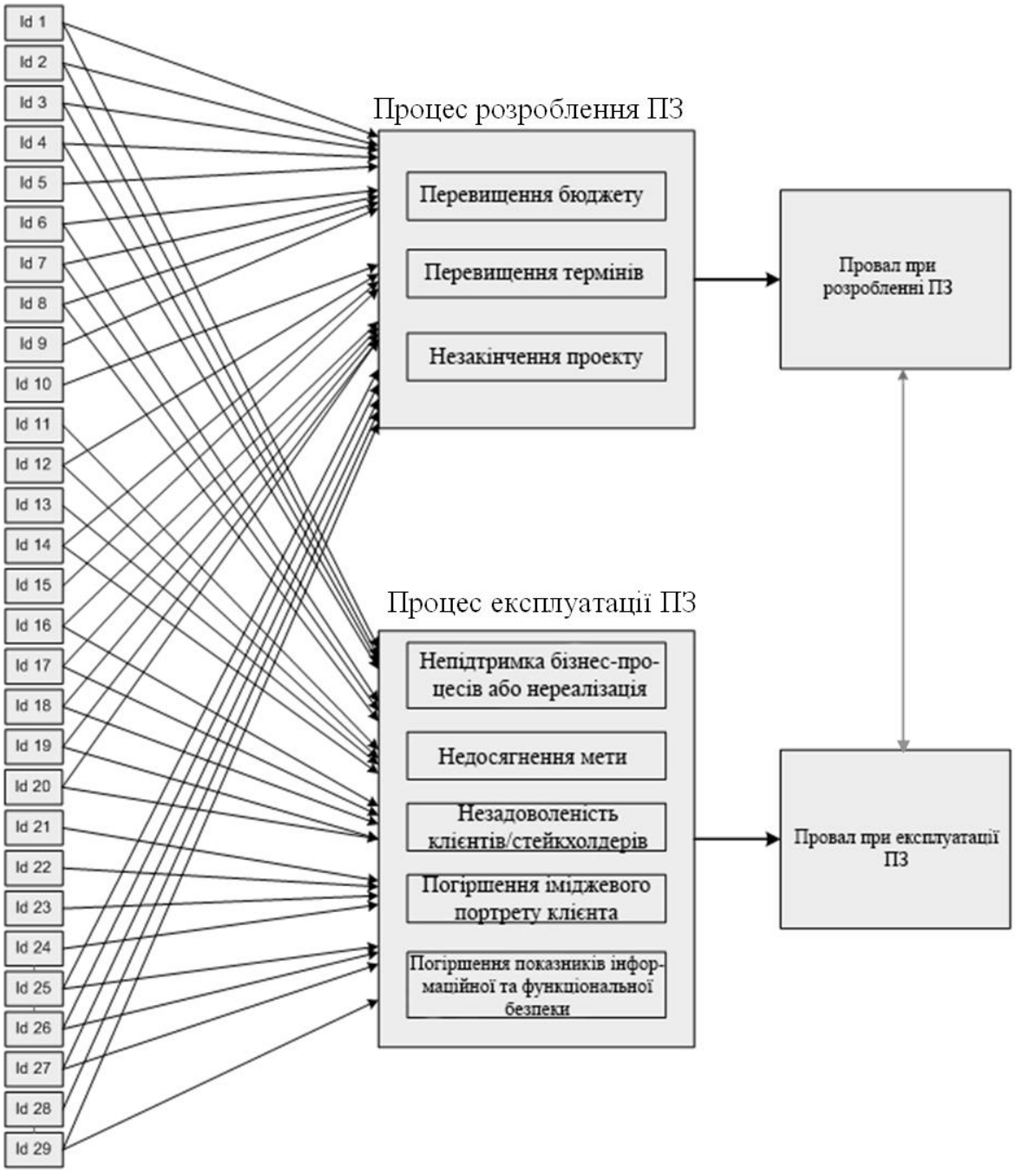


Рис. 2.4. Схема впливу вразливостей на основні фактори успіху розроблення, впровадження і тривалої експлуатації ПЗ

Відповідно до відомого принципу Парето [34], серед множини потенційних причин (причинних факторів, за Ішикавою), що породжують

проблеми (наслідки), лише дві-три є найбільш значущими, їх пошук і має бути організований. Для цього здійснюється:

– збір і систематизація усіх причин, що прямо чи опосередковано впливають на досліджувану проблему;

– групування цих причин за смисловими та причинно-наслідковими блоками;

– ранжирування їх усередині кожного блоку;

– аналіз отриманої картини.

Тому цей інструмент дозволяє прояснити і врахувати усі істотні фактори, що впливають на результат розроблення ПЗ.

Застосування діаграми Ішикави дозволяє з'ясувати причини будь-яких проблем в організації або, наприклад, причини виникнення експлуатаційних "багів" ПЗ. При цьому діаграма Ішикави має ряд переваг:

– допомагає наочно показати зв'язки між отриманим результатом і причинами, що викликали його;

– дозволяє провести аналіз ланцюжка факторів, що впливають на проблему.

Розглянемо загальні правила побудови діаграми Ішикави.

1. Для початку всі учасники проекту разом мають сформулювати проблему для можливості розпочати першочергову стадію побудову діаграми.

2. Спочатку рисують основну задачу (проблему що досліджується) у вигляді прямокутника з правого боку аркуша по середині, до якої з лівої сторони підводиться основна горизонтальна лінія.

3. Після цього наносяться головні причини (причини рівня 1), що впливають на проблему. Вони з'єднуються похилими стрілками з основною горизонтальною лінією.

4. Дописуються причини другого рівня, які впливають на головні причини. Наносяться примикаючи до "більших" причин другого рівня. Причини третього рівня впливають на причини другого рівня і т.д.. Якщо нанесені не всі причини на діаграмі, то стрілка залишається без надпису.

5. Так як метою діаграми є пошук оптимального шляху та найефективніше рішення проблеми, то аналізуючи необхідно виявляти та фіксувати навіть самі незначні фактори. Оцінку та сортування причин-факторів проводять по значимості, при чому відзначають найважливіші, які впливають на показник якості. Використовуючи запропонований алгоритм і враховуючи вразливості, описані вище на рис. 2.3, 2.4, діаграму Ішикави можна представити у вигляді рис. 2.5.

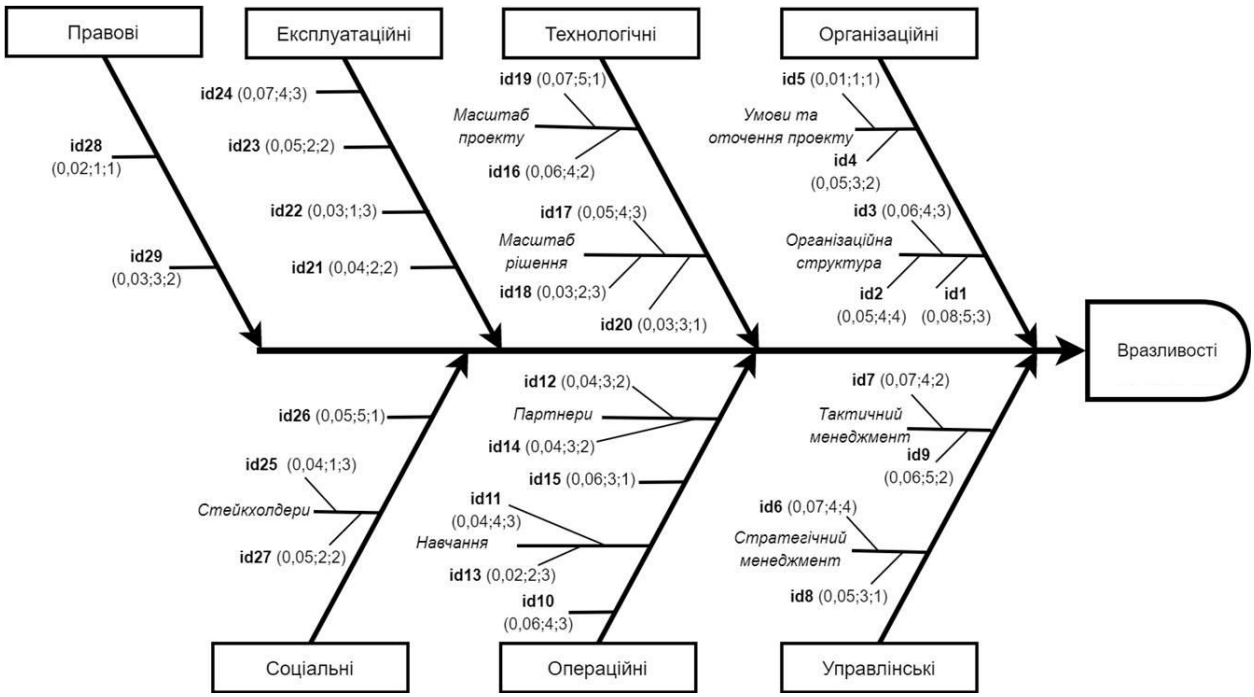


Рис. 2.5. Дерево рішень діаграми вразливостей розроблення ПЗ

У результаті застосування запропонованого алгоритму приклад обраних параметрів оцінки вразливостей та ймовірність виникнення вразливості наведено у табл. 2.1.

Таблиця 2.1. Обрані параметри оцінки вразливостей розроблення ПЗ

| № | Ймовірність виникнення вразливості | Обрані параметри оцінки вразливостей | | № | Ймовірність виникнення вразливості | Обрані параметри оцінки вразливостей | |
|---|---|---|---|---|---|---|---|
| | | Збитки економічні (5-1) | Збитки репутації фірми (4-1) | | | Збитки економічні (5-1) | Збитки репутації фірми (4-1) |
| Id1 | 0,08 | 5 | 3 | Id16 | 0,06 | 4 | 2 |
| Id2 | 0,05 | 4 | 4 | Id17 | 0,05 | 4 | 3 |
| Id3 | 0,06 | 4 | 3 | Id18 | 0,03 | 2 | 3 |
| Id4 | 0,05 | 3 | 2 | Id19 | 0,07 | 5 | 1 |
| Id5 | 0,01 | 1 | 1 | Id20 | 0,03 | 3 | 1 |
| Id6 | 0,07 | 4 | 4 | Id21 | 0,04 | 2 | 2 |
| Id7 | 0,07 | 4 | 2 | Id22 | 0,03 | 1 | 3 |
| Id8 | 0,05 | 3 | 1 | Id23 | 0,05 | 2 | 2 |
| Id9 | 0,06 | 5 | 2 | Id24 | 0,07 | 4 | 3 |
| Id10 | 0,06 | 4 | 3 | Id25 | 0,04 | 1 | 3 |
| Id11 | 0,04 | 4 | 3 | Id26 | 0,05 | 5 | 1 |
| Id12 | 0,04 | 3 | 2 | Id27 | 0,05 | 2 | 2 |
| Id13 | 0,02 | 2 | 3 | Id28 | 0,02 | 1 | 1 |
| Id14 | 0,04 | 3 | 2 | Id29 | 0,03 | 3 | 2 |
| Id15 | 0,06 | 3 | 1 | | | | |

Зображення діаграми Ішикави дає можливість отримати детальнішу інформацію про можливість взаємовпливу різних видів вразливості один на одного, що так само дасть уточнювальні дані для кількісного аналізу вразливостей. Проте, завдання вибору найбільш пріоритетних вразливостей діаграма вирішити не може.

Для вирішення цього завдання в дисертаційній роботі пропонується використати математичний апарат багатокритеріальної оптимізації, заснованої на локальній геометрії множини Парето.

Аналіз літератури [34, 35] показав, що існують, принаймні, три формулювання багатокритеріальної оптимізації, заснованої на локальній геометрії множини Парето:

1. Локальна. Знайти одне Парето-оптимальне рішення (найближче до заданої початкової точки).

2. Глобальна. Знайти скінченну множину Парето-оптимальних рішень, що досить добре описує (покриває) істинний Парето-фронт.

3. Інтерактивна. Знайти Парето-оптимальне рішення, що максимально задовольняє перевагам менеджера, який приймає рішення (МПР).

Проведені дослідження показали, що в процесах, побудованих на принципах постійних комунікацій між учасниками, використання "мозкових штурмів" із залученням думок експертів, доцільним є використання інтерактивного формулювання багатокритеріальної оптимізації.

У цих умовах абстрактне завдання вибору найбільш важливих вразливостей розроблення ПЗ з наявної початкової множини можливих (допустимих) варіантів (рішень) $X$ можна сформулювати наступним чином.

Позначимо множину усіх заздалегідь визначених вразливостей розроблення ПЗ через $S(X)$. Очевидно, $S(X) \subset X$.

Таким чином, в завданні вибору дана множина $X$, що містить, принаймні, два елементи, а вимагається знайти деяку його непорожню підмножину $S(X)$. Передбачається, що вибір робиться МПР, в ролі якого може виступати як окрема людина, так і цілий колектив розробників.

Для того, щоб здійснюваний вибір найбільшою мірою відповідав досягненню наявної мети (тобто був "найкращим" або "оптимальнішим" для цього МПР), необхідно в процесі вибору зважати на думку експертів.

Проведені дослідження показали, що нині існує багато підходів до врахування думки експертів (метод аналізу ієрархій [35], реалізований в програмному продукті *EXPERT CHOICE*, метод "штучного" відношення

переваги [36], та ін.) проте усі вони мають істотні недоліки, головний з яких полягає в тому, що, незважаючи на різноманіття і детальну вивченість ієрархій і "штучних" взаємовідношень, украй рідкісне яке-небудь з них можна вважати таким, що задовольняє конкретного МПР.

Характерним прикладом, що підтверджує цей факт, являється нехтування процесом оцінки вразливостей розробленого ПЗ (недостатність або повна відсутність pen-тестування).

Тому для вирішення завдання вибору найбільш пріоритетних вразливостей (звуження множини Парето) пропонується використати "кванти інформації". Для цього розглянемо довільні оцінки вразливостей розроблення ПЗ $y'=(y_1',\ldots,y_m')$ і $y''=(y_1'',\ldots,y_m'')$, що належать до множини парето-оптимальних векторів $f(P_f(X))$. За визначенням множини Парето, повинні знайтися такі дві непорожні підмножини номерів критеріїв $A,B\subset I=\{1,2,\ldots,m\}$, що

$$y_i'>y_i'',\quad y_i'-y_i''=w_i>0,\quad \forall i\in A \tag{2.1}$$

$$y_j''>y_j',\quad y_j''-y_j'=w_j>0,\quad \forall j\in B \tag{2.2}$$

$$y_s''=y_s',\quad \forall s\in I\setminus(A\cup B) \tag{2.3}$$

Згідно з умовами (2.1-2.3), перший вектор перевершує другий за компонентами групи критеріїв А, тоді як другий перевершує перший за компонентами групи критеріїв В. За іншими компонентами (якщо такі є) два вказані вектори співпадають. Звуження множини Парето, тобто видалення деяких парето-оптимальних векторів, зазвичай відбувається на основі порівняння.

Людині найпростіше порівнювати пари. Якщо при порівнянні фіксованої пари парето-оптимальних векторів у′ і у″ виду (2.1-2.3) МПР "вибраковує" один з цих векторів (наприклад, другий), то це означає, що для нього перший вектор прийнятніший за другий, тобто $y'\succ y''$, де $\succ$ –

відношення переваги, визначене на усьому критеріальному просторі $\Re^m$ і співпадаюче на множині $Y$ з відношенням $\succ y$.

Співвідношення $y' \succ y''$, задає "квант інформації" про відношення строгої переваги, який свідчить про готовність МПР до компромісу − він згоден піти на втрати за усіма критеріями групи В в розмірі $w_j$ заради того, щоб отримати надбавки в розмірі $w_i$ за критеріями групи А, зберігши при цьому значення усіх інших критеріїв.

Наявність вказаного "кванта інформації" дозволяє скоротити множину Парето на один вектор $y''$. Для того, щоб добитися більшого скорочення, можна прийняти, що $y' \succ y''$, має місце не лише для даної пари векторів, але і для всіх тих векторів, які задовольняють умовам (2.1-2.3) при незмінних значеннях $w_i$ і $w_j$.

В цьому випадку пропонується говорити, що група критеріїв А важливіша за групу В. При вказаному розширенні дії "кванта інформації" можна розраховувати на помітніше звуження множини Парето, хоча нерідко і воно виявляється недостатнім для остаточного вибору. У таких випадках має сенс накласти додаткові вимоги на відношення переваги так, щоб дія "кванта інформації" в звуженні множини Парето виявилася ефективнішою. Ці вимоги (без аксіоми виключення) сформульовані в [34]. Пізніше було встановлено, що вони являють собою подальшие посилення системи двох згадуваних раніше аксіом, що гарантують виконання принципу Еджворта-Парето.

Аксіома 1 (аксіома виключення).

Аксіома 2. Відношення $\succ$ визначене на усьому критерійному просторі $\Re^m$ і є транзитивним на ньому.

Аксіома 3 (аксіома узгодження). З двох векторів, що відрізняються один від одного єдиною компонентою, для МПР є прийнятнішим вектор, що має велику компоненту.

Аксіома 4 (аксіома інваріантності). Відношення переваги $\succ$ інваріантне відносно лінійного позитивного перетворення (тобто є лінійним).

Нехай один критерій (чи група критеріїв) важливіший за інший критерій (іншої групи критеріїв), якщо має місце деяка умова $\Xi$, яка містить певну інформацію про відношення переваги МПР. Звідси зрозуміло, що без визначення важливості критеріїв завжди можна обійтися, оперуючи в процесі прийняття рішень безпосередньо з умовою $\Xi$. Щоб скористатися визначенням важливості, заснованим на "кванті інформації" і використовуючим як умову $\Xi$ співвідношення (2.1-2.3), спочатку слід пояснити МПР це визначення важливості, переконатися, що він його "засвоїв", після чого для виявлення переваг МПР поставити йому питання на "мові важливості": чи є група критеріїв A важливішою за групу B з параметрами $w_i$ і $w_j$ (для $i \in A$ і $j \in B$).

З [37] відомо, що бінарне відношення $\prec$, яке задане на векторному просторі $\Re^m$, називається конусним, якщо існує такий конус $K \subset \Re^m$, що співвідношення $y' \succ y''$ має місце тоді і лише тоді, коли $y' - y'' \in K$.

Аксіома 5. Будь-яке бінарне відношення $\prec$, задане на векторному просторі $\Re^m$ і задовольняюче аксіомам 2−4, є конусним з гострим опуклим конусом (без початку координат), який містить усі вектори з ненегативними компонентами. Тобто будь-яке конусне відношення $\prec$ з вказаним конусом задовольняє аксіомам 2−4.

Аксіома 5 відкриває можливість використання апарату опуклого аналізу і побудови змістовної математичної теорії для врахування різного набору "квантів інформації". Найбільш простий випадок одного "кванта" розглядається в наступному твердженні, доказ якого спирається на факти з теорії двоїстості опуклого аналізу.

Аксіома 6. Нехай виконані аксіоми 2-4 і є "квант інформації" про відношення переваги $\prec$. Тоді для будь-якої множини обираних варіантів

*S(X)*, що задовольняє аксіомі 1, справедливі включення $S(X) \subset P_g(X) \subset P_f(X)$, причому "новий" векторний критерій $g$ може бути утворений з функцій $f_i$ для всіх $i \in I \setminus B$:

$$g_{i,j} = w_j f_i + w_i f_j \text{ для всіх } i \in A, j \in B, \tag{2.4}$$

або з функцій $f_i$ для всіх $i \in I \setminus B$:

$$f_0 = \min_{i \in A} \frac{f_i}{w_i} + \min_{j \in B} \frac{f_j}{w_j} \tag{2.5}$$

Важлива особливість аксіоми 6 полягає у відсутності яких-небудь вимог до множини *X* і векторному критерію *f*: ці об'єкти можуть бути будь-якими. Обмеження накладаються лише на поведінку МПР в процесі прийняття рішень і виражаються вони у формі аксіом 1-4.

Аксіомою 6 вказується оцінка згори $P_g(X)$ для невідомої множини вибираних варіантів *S(X)*, більш точна, ніж множина Парето $P_f(X)$. Сама оцінка є множиною парето-оптимальних варіантів, але відносно "нового" векторного критерію *g*.

Для того, щоб сформувати *g*, зі "старого" векторного критерію *f* слід видалити всі компоненти групи критеріїв B і додати один нелінійний критерій $f_0$ виду (2.5), або $|A| \cdot |B|$ "нових" лінійних критеріїв виду (2.4), де $|L|$ означає число елементів скінченної множини *L*.

Варіант з нелінійною функцією $f_0$ виду (2.5) можна застосовувати для кількісних критеріїв, значення яких вимірюються в шкалі відносин, тоді як варіант (2.4) допускає використання ще і в шкалі інтервалів.

Нелінійну функцію $f_0$ виду (2.5) можна використати для вивчення випадку, коли одна група критеріїв важливіша за іншу, де на відміну від приведеної вище аксіоматики використовується операція транзитивного замикання бінарного відношення і деякі інші припущення.

Як показали дослідження, врахування декількох "квантів інформації" повинно більшою мірою сприяти звуженню множини Парето. Проте, інколи може статися ситуація, коли ряд "квантів інформації" матиме суперечливий сенс, і їх використання буде неможливим. Тому важливим є завдання вибору несуперечливих "квантів інформації". У рамках дисертаційної роботи під несуперечливою названа така множина, яка "породжує" іррефлексивне відношення.

Побудова оцінки зверху для невідомої множини вибираних векторів *S(Y)=f(S(X))* у вигляді множини $\overline{P}(Y) = f\left(P_g(X)\right)$ за наявності довільного несуперечливого кінцевого набору "квантів інформації" у разі скінченної множини *Y* зводиться до послідовної перевірки співвідношення

$$y' \succ_m y'' \qquad (2.6)$$

для усіх пар допустимих векторних оцінок $y', y'' \in Y$, де $\succ_m$ - бінарне відношення, яке будується на основі наявної несуперечливої множини "квантів інформації".

Таким чином, отримано подальший розвиток структурної ідентифікації вразливостей розроблення ПЗ, що відрізняється від відомих побудовою оцінки вразливостей розроблення ПЗ "зверху" у вигляді множини, за наявності довільного несуперечливого кінцевого набору "квантів інформації". Використовуючи приведений вище метод проведемо оцінку рангу вразливостей розроблення ПЗ.

### 2.2.3 Приклад розрахунку даних ітерації проекту, методом якісного аналізу вразливостей розроблення ПЗ

Розглянемо приклад використання розроблених методів якісного аналізу та кількісної оцінки вразливостей розроблення ПЗ.

В якості прикладу розглянемо одну з ітерацій (sprint) розроблення ПЗ замовлення таксі за допомогою однієї з гнучких методологій розроблення ПЗ – SCRUM (рис 2.6).

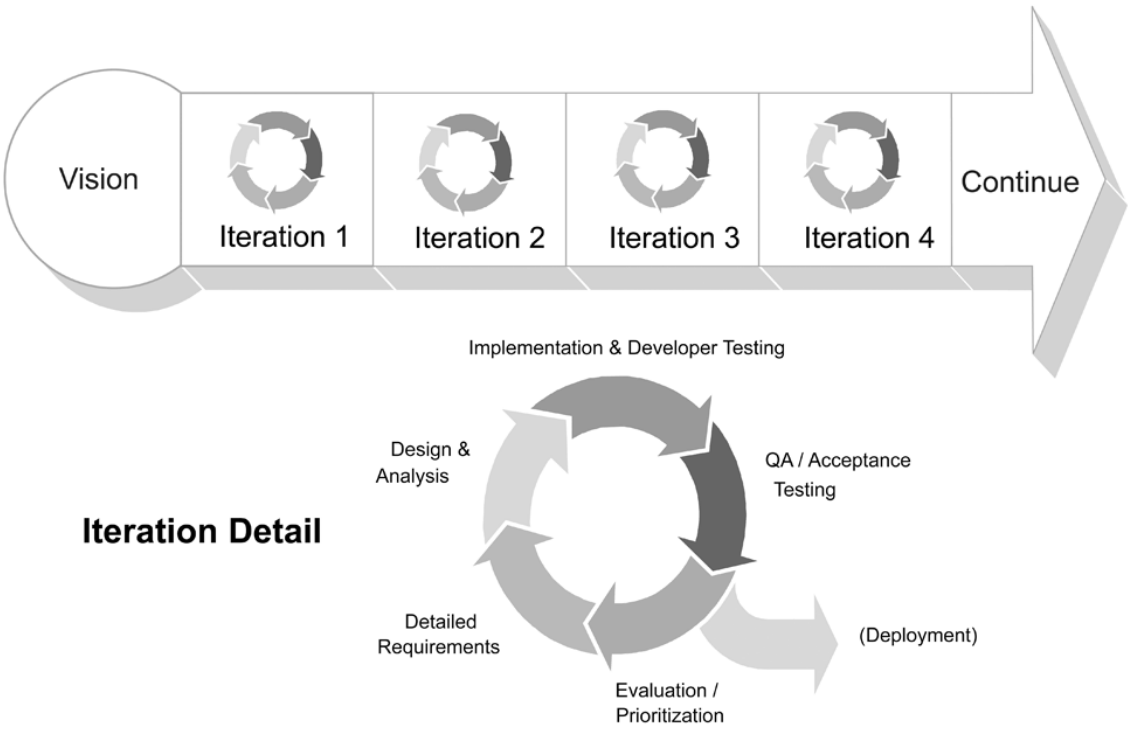


Рис. 2.6. Загальна структура гнучкої методології SCRUM

Scrum це фреймворк управління, згідно з яким одна чи кілька кроссфункціональних самоорганізованих команд розробляють ПЗ поетапно.

У Scrum є система ролей, зустрічей та правил. У цій моделі розроблення ПЗ за створення і адаптацію робочих процесів відповідають команди.

У Scrum використовуються ітерації фіксованої довжини, так звані спринти. Вони зазвичай займають 1-2 тижні (до 1 місяця). Scrum команди прагнуть створити готовий до впровадження (якісно протестоване) ПЗ.

Scrum не лінійний метод розроблення ПЗ; це не каскадна модель. Каскадна модель - лінійна послідовність подій, коли ПЗ планують,

розробляють, тестують і так далі. Жоден її етап не можна починати, поки не завершений попередній.

За Scrum, продукт розробляють не відразу повністю, а невеликими готовими до релізу частинами, кожну з яких завершують за коротку ітерацію.

Короткі вхідні дані прикладу:

1. Назва проекту – Airplane tickets.

2. Короткий опис проекту – замовлення авіаквитків різних авіакомпаній. Сьома ітерація розроблення.

3. Склад команди розробників – 4 розробника ПЗ, 1 Scrum Master, 2 дизайнери інтерфейсу, 1 тестувальник ПЗ.

4. Обрана методологія розроблення ПЗ – гнучка методологія розроблення ПЗ SCRUM.

5. Обраний сталий набір параметрів оцінки вразливостей для проекту: Ймовірність виникнення вразливості; Збитки економічні; Збитки репутаційні.

Після початкового обговорення, планування та отримання бєклогу ітерації (sprint backlog) – отримання поточного технічного завдання. Скрам майстер (Scrum Master) на першій щоденній зустрічі ітерації (Daily Scrum Meeting) з колективом розробників за допомогою методу мозкового штурму (brainstorming) та діаграми Ішикави обговорює, редагує та знаходить залежності для кожної вразливості розроблення ПЗ (рис. 2.3).

Огляд обраного набору параметрів:

1. Ймовірність виникнення вразливості.

2. Збитки економічні. Розподіляються на п’ять можливих наслідків у результаті виникнення: 1 – незначний; 2 – середній; 3 – важкий; 4 – дуже важкий, погано прогнозований; 5 – критичний, крах розроблення ПЗ, неможливість подальшої розробки.

3. Збитки репутації фірми. Розподіляються на чотири можливих наслідки у результаті виникнення: 1 – незначна; 2 – середня, простої

розробників (зняття з завдань по розробці); 3 – висока недовіра, втрата майбутніх замовлень проектів; 4 – критична, прогнозована повна втрата довіри, неможливе подальше співробітництво.

Параметри вибираються у відповідності з експертною оцінкою всієї команди, опираючись на базу знань фірми, маркетингові та наукові дані з відкритих джерел та документації замовника ПЗ. Отримані результати представлено в табл. 2.1, це вхідні дані ітерації (sprint) розроблення інтерфейсу ПЗ.

Розглянемо кроки алгоритму якісного аналізу вразливостей розроблення ПЗ на основі математичного апарату багатокритеріальної оптимізації, заснованої на локальній геометрії множини Парето (розділ 2.2.2).

Крок 1. Розрахунок множини Паретто.

На рис. 2.7-2.15 наведений приклад розрахунку множини Паретто відповідно вхідних даних табл. 2.1. Для наглядності візуалізації розрахунку прикладу отримані дані спочатку сортуються по сумі коефіцієнтів векторів. Та формується множина Паретто оптимальних рішень. Ітерація 1. Парето-оптимальне рішення id1, id6. Ітерація 2, парето-оптимальне рішення id24, id2, id9, id19. Ітерація 3, парето-оптимальне рішення id10, id3, id7, id26. Ітерація 4, парето-оптимальне рішення id17, id16. Ітерація 5, парето-оптимальне рішення id11, id4, id15. Ітерація 6, парето-оптимальне рішення id12, id14, id23, id27, id8. Ітерація 7, парето-оптимальне рішення id18, id25, id29, id21. Ітерація 8, парето-оптимальне рішення id22, id20, id13. Ітерація 9, парето-оптимальне рішення id28. Ітерація 10, парето-оптимальне рішення id5.

У результаті сформована множина Паретто оптимальних рішень представлена у табл. 2.2.

Крок 2. Отримання квантів інформації, розрахунок скалярного критерію звуження початкової множини Парето.

В процесах, побудованих на принципах постійних комунікацій між учасниками, використання "мозкових штурмів" із залученням думок експертів, доцільним є використання інтерактивного формулювання багатокритеріальної оптимізації.

Після отримання множини Паретто оптимальних рішень Скрам майстер (Scrum Master) на щоденній зустрічі ітерації (Daily Scrum Meeting) за допомогою методу мозгового штурма (brainstorming) проводить вибор найбільш пріоритетних вразливостей розроблення ПЗ з наявної початкової множини можливих (допустимих) варіантів (рішень) рисунок 2.16.

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Id 1 | Id 6 | Id 24 | Id 10 | Id 3 | Id 2 | Id 17 | Id 9 | Id 7 | Id 16 | Id 19 | Id 11 | Id 26 | Id 4 | Id 15 | Id 12 | Id 23 | Id 14 | Id 27 | Id 25 | Id 18 | Id 8 | Id 29 | Id 13 | Id 22 | Id 21 | Id 20 | Id 28 | Id 5 | |
| Id1 | 0,08 | 5 | 3 | 0 | Id1 | 3 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 82 |
| Id6 | 0,07 | 4 | 4 | 1 | Id6 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 79 |
| Id24 | 0,07 | 4 | 3 | 2 | Id24 | 1 | 2 | 3 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 77 |
| Id10 | 0,06 | 4 | 3 | 3 | Id10 | 1 | 1 | 2 | 3 | 3 | 2 | 3 | 2 | 2 | 3 | 1 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 73 |
| Id3 | 0,06 | 4 | 3 | 4 | Id3 | 1 | 1 | 2 | 3 | 3 | 2 | 3 | 2 | 2 | 3 | 1 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 73 |
| Id2 | 0,05 | 4 | 4 | 5 | Id2 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | 1 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 70 |
| Id17 | 0,05 | 4 | 3 | 6 | Id17 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 1 | 2 | 2 | 1 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 68 |
| Id9 | 0,06 | 5 | 2 | 7 | Id9 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 67 |
| Id7 | 0,07 | 4 | 2 | 8 | Id7 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 67 |
| Id16 | 0,06 | 4 | 2 | 9 | Id16 | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 63 |
| Id19 | 0,07 | 5 | 1 | 10 | Id19 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 61 |
| Id11 | 0,04 | 4 | 3 | 11 | Id11 | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 3 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 61 |
| Id26 | 0,05 | 5 | 1 | 12 | Id26 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 52 |
| Id4 | 0,05 | 3 | 2 | 13 | Id4 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 49 |
| Id15 | 0,06 | 3 | 1 | 14 | Id15 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 44 |
| Id12 | 0,04 | 3 | 2 | 15 | Id12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 42 |
| Id23 | 0,05 | 2 | 2 | 16 | Id23 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 3 | 42 |
| Id14 | 0,04 | 3 | 2 | 17 | Id14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 42 |
| Id27 | 0,05 | 2 | 2 | 18 | Id27 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 3 | 42 |
| Id25 | 0,04 | 1 | 3 | 19 | Id25 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 3 | 2 | 1 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 40 |
| Id18 | 0,03 | 2 | 3 | 20 | Id18 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 3 | 1 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 40 |
| Id8 | 0,05 | 3 | 1 | 21 | Id8 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 39 |
| Id29 | 0,03 | 3 | 2 | 22 | Id29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 37 |
| Id13 | 0,02 | 2 | 3 | 23 | Id13 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 1 | 1 | 3 | 2 | 2 | 1 | 3 | 3 | 36 |
| Id22 | 0,03 | 1 | 3 | 24 | Id22 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 3 | 1 | 2 | 3 | 3 | 35 |
| Id21 | 0,04 | 2 | 2 | 25 | Id21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 3 | 35 |
| Id20 | 0,03 | 3 | 1 | 26 | Id20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 3 | 3 | 3 | 27 |
| Id28 | 0,02 | 1 | 1 | 27 | Id28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 3 | 3 | 11 |
| Id5 | 0,01 | 1 | 1 | 28 | Id5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 2 | 3 | 9 |

Рис. 2.7. Ітерація 1, Парето-оптимальне рішення id1, id6

Крок 3. Звуження початкової множини Парето на основі отриманих даних «квантів інформації».

Інформація про відношення переваги, задана отриманими чотирма «квантами інформації» перевірена та є несуперечливою.

Застосуємо описаний алгоритм (розділ 2.2.2) для формування нового векторного критерію.

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Id 24 | Id 10 | Id 3 | Id 2 | Id 17 | Id 9 | Id 7 | Id 16 | Id 11 | Id 19 | Id 26 | Id 4 | Id 15 | Id 12 | Id 23 | Id 14 | Id 27 | Id 25 | Id 18 | Id 8 | Id 29 | Id 13 | Id 21 | Id 22 | Id 20 | Id 28 | Id 5 | |
| Id24 | 0,07 | 4 | 3 | 0 | Id24 | 3 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 77 |
| Id10 | 0,06 | 4 | 3 | 1 | Id10 | 2 | 3 | 3 | 2 | 3 | 2 | 2 | 3 | 3 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 74 |
| Id3 | 0,06 | 4 | 3 | 2 | Id3 | 2 | 3 | 3 | 2 | 3 | 2 | 2 | 3 | 3 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 74 |
| Id2 | 0,05 | 4 | 4 | 3 | Id2 | 2 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | 3 | 1 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 70 |
| Id17 | 0,05 | 4 | 3 | 4 | Id17 | 2 | 2 | 2 | 2 | 3 | 1 | 2 | 2 | 3 | 1 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 69 |
| Id9 | 0,06 | 5 | 2 | 5 | Id9 | 1 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 68 |
| Id7 | 0,07 | 4 | 2 | 6 | Id7 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 68 |
| Id16 | 0,06 | 4 | 2 | 7 | Id16 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 65 |
| Id11 | 0,04 | 4 | 3 | 8 | Id11 | 2 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 3 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 62 |
| Id19 | 0,07 | 5 | 1 | 9 | Id19 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 61 |
| Id26 | 0,05 | 5 | 1 | 10 | Id26 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 53 |
| Id4 | 0,05 | 3 | 2 | 11 | Id4 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 52 |
| Id15 | 0,06 | 3 | 1 | 12 | Id15 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 47 |
| Id12 | 0,04 | 3 | 2 | 13 | Id12 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 45 |
| Id23 | 0,05 | 2 | 2 | 14 | Id23 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 45 |
| Id14 | 0,04 | 3 | 2 | 15 | Id14 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 45 |
| Id27 | 0,05 | 2 | 2 | 16 | Id27 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 45 |
| Id25 | 0,04 | 1 | 3 | 17 | Id25 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 3 | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 3 | 42 |
| Id18 | 0,03 | 2 | 3 | 18 | Id18 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 3 | 1 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 42 |
| Id8 | 0,05 | 3 | 1 | 19 | Id8 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 42 |
| Id29 | 0,03 | 3 | 2 | 20 | Id29 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 40 |
| Id13 | 0,02 | 2 | 3 | 21 | Id13 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 1 | 1 | 3 | 2 | 2 | 1 | 3 | 3 | 38 |
| Id22 | 0,03 | 1 | 3 | 22 | Id21 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 38 |
| Id21 | 0,04 | 2 | 2 | 23 | Id22 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 3 | 2 | 3 | 3 | 37 |
| Id20 | 0,03 | 3 | 1 | 24 | Id20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 3 | 3 | 3 | 30 |
| Id28 | 0,02 | 1 | 1 | 25 | Id28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 3 | 3 | 14 |
| Id5 | 0,01 | 1 | 1 | 26 | Id5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 12 |

Рис. 2.8. Ітерація 2, Парето-оптимальне рішення id24, id2, id9, id19

Розглянемо векторні добутки всіх отриманих пар векторів, які до решти набору мають позитивні скалярні добутки.

Не важко переконатися, що критерієм відбору задовольняє тільки один вектор $y_1 = \{2;0{,}01;0{,}01\}$. В результаті знайдений один вектор $y_1$. Йому відповідає новий скалярний критерій g з компонентами $g(x) = 2x_1 + 0.01x_2 + 0.01x_3$. Відповідно до сформульованої вище теореми, множина Парето щодо цього 1-мірного критерію буде точнішою оцінкою для множини обраних векторів варіантів у вихідній множині Парето.

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Id 10 | Id 3 | Id 17 | Id 7 | Id 16 | Id 11 | Id 4 | Id 26 | Id 15 | Id 12 | Id 14 | Id 23 | Id 27 | Id 8 | Id 18 | Id 25 | Id 29 | Id 21 | Id 13 | Id 22 | Id 20 | Id 28 | Id 5 | |
| Id10 | 0,06 | 4 | 3 | 0 | Id10 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 67 |
| Id3 | 0,06 | 4 | 3 | 1 | Id3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 67 |
| Id17 | 0,05 | 4 | 3 | 2 | Id17 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 63 |
| Id7 | 0,07 | 4 | 2 | 3 | Id7 | 2 | 2 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 60 |
| Id16 | 0,06 | 4 | 2 | 4 | Id16 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 59 |
| Id11 | 0,04 | 4 | 3 | 5 | Id11 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 1 | 2 | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 57 |
| Id4 | 0,05 | 3 | 2 | 6 | Id4 | 0 | 0 | 1 | 1 | 1 | 1 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 49 |
| Id26 | 0,05 | 5 | 1 | 7 | Id26 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 47 |
| Id15 | 0,06 | 3 | 1 | 8 | Id15 | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 44 |
| Id12 | 0,04 | 3 | 2 | 9 | Id12 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 43 |
| Id14 | 0,04 | 3 | 2 | 10 | Id14 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 43 |
| Id23 | 0,05 | 2 | 2 | 11 | Id23 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 42 |
| Id27 | 0,05 | 2 | 2 | 12 | Id27 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 42 |
| Id8 | 0,05 | 3 | 1 | 13 | Id8 | 0 | 0 | 1 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 40 |
| Id18 | 0,03 | 2 | 3 | 14 | Id18 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 3 | 2 | 2 | 2 | 3 | 3 | 2 | 3 | 3 | 39 |
| Id25 | 0,04 | 1 | 3 | 15 | Id25 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 3 | 2 | 2 | 2 | 3 | 2 | 3 | 3 | 39 |
| Id29 | 0,03 | 3 | 2 | 16 | Id29 | 0 | 0 | 0 | 1 | 1 | 0 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 38 |
| Id21 | 0,04 | 2 | 2 | 17 | Id21 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 36 |
| Id13 | 0,02 | 2 | 3 | 18 | Id13 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 1 | 3 | 3 | 35 |
| Id22 | 0,03 | 1 | 3 | 19 | Id22 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 3 | 2 | 3 | 3 | 34 |
| Id20 | 0,03 | 3 | 1 | 20 | Id20 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 3 | 3 | 3 | 29 |
| Id28 | 0,02 | 1 | 1 | 21 | Id28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 3 | 3 | 13 |
| Id5 | 0,01 | 1 | 1 | 22 | Id5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 11 |

Рис. 2.9. Ітерація 3, Парето-оптимальне рішення id10, id3, id7, id26

Для додаткового звуження множин Парето використовуємо новий знайдений критерій $g(x)$. Нехай маємо $X = \{Id1, Id6\}$, або що те ж саме: $X = \{(0{,}08, 5, 3), (0{,}07, 4, 4)\}$. Використавши критерій $g(x)$ отримаємо $Y = \{0{,}08 \cdot 2 + 5 \cdot 0{,}01 + 3 \cdot 0{,}01;\ 0{,}07 \cdot 2 + 4 \cdot 0{,}01 + 4 \cdot 0{,}01\}$, зробивши обчислення:

Y={0,24; 0,22}. Відповідно отриманим вагам 0,24>0,22, тому Id1 ≻ Id6 – множина Парето {Id1; Id6} розділена на дві {Id1} та {Id6} з належністю по одному елементу.

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 Id 17 | 1 Id 16 | 2 Id 11 | 3 Id 4 | 4 Id 12 | 5 Id 14 | 6 Id 15 | 7 Id 23 | 8 Id 27 | 9 Id 8 | 10 Id 29 | 11 Id 18 | 12 Id 25 | 13 Id 21 | 14 Id 13 | 15 Id 22 | 16 Id 20 | 17 Id 28 | 18 Id 5 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id17 | 0,05 | 4 | 3 | 0 | Id17 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 55 |
| Id16 | 0,06 | 4 | 2 | 1 | Id16 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 51 |
| Id11 | 0,04 | 4 | 3 | 2 | Id11 | 2 | 2 | 3 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 50 |
| Id4 | 0,05 | 3 | 2 | 3 | Id4 | 1 | 1 | 1 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 46 |
| Id12 | 0,04 | 3 | 2 | 4 | Id12 | 0 | 1 | 1 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 41 |
| Id14 | 0,04 | 3 | 2 | 5 | Id14 | 0 | 1 | 1 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 41 |
| Id15 | 0,06 | 3 | 1 | 6 | Id15 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 40 |
| Id23 | 0,05 | 2 | 2 | 7 | Id23 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 39 |
| Id27 | 0,05 | 2 | 2 | 8 | Id27 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 39 |
| Id8 | 0,05 | 3 | 1 | 9 | Id8 | 1 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 38 |
| Id29 | 0,03 | 3 | 2 | 10 | Id29 | 0 | 1 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 36 |
| Id18 | 0,03 | 2 | 3 | 11 | Id18 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 2 | 3 | 3 | 2 | 3 | 3 | 35 |
| Id25 | 0,04 | 1 | 3 | 12 | Id25 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 3 | 3 | 35 |
| Id21 | 0,04 | 2 | 2 | 13 | Id21 | 0 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 34 |
| Id13 | 0,02 | 2 | 3 | 14 | Id13 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 1 | 3 | 3 | 31 |
| Id22 | 0,03 | 1 | 3 | 15 | Id22 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 3 | 2 | 3 | 3 | 30 |
| Id20 | 0,03 | 3 | 1 | 16 | Id20 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 28 |
| Id28 | 0,02 | 1 | 1 | 17 | Id28 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 3 | 3 | 12 |
| Id5 | 0,01 | 1 | 1 | 18 | Id5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 2 | 3 | 10 |

Рис. 2.10. Ітерація 4, Парето-оптимальне рішення id17, id16

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 Id 11 | 1 Id 4 | 2 Id 12 | 3 Id 14 | 4 Id 15 | 5 Id 23 | 6 Id 27 | 7 Id 8 | 8 Id 29 | 9 Id 18 | 10 Id 25 | 11 Id 21 | 12 Id 13 | 13 Id 22 | 14 Id 20 | 15 Id 28 | 16 Id 5 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id11 | 0,04 | 4 | 3 | 0 | Id11 | 3 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 46 |
| Id4 | 0,05 | 3 | 2 | 1 | Id4 | 1 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 44 |
| Id12 | 0,04 | 3 | 2 | 2 | Id12 | 1 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 40 |
| Id14 | 0,04 | 3 | 2 | 3 | Id14 | 1 | 2 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 40 |
| Id15 | 0,06 | 3 | 1 | 4 | Id15 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 38 |
| Id23 | 0,05 | 2 | 2 | 5 | Id23 | 1 | 2 | 2 | 2 | 1 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 37 |
| Id27 | 0,05 | 2 | 2 | 6 | Id27 | 1 | 2 | 2 | 2 | 1 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 37 |
| Id8 | 0,05 | 3 | 1 | 7 | Id8 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 37 |
| Id29 | 0,03 | 3 | 2 | 8 | Id29 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 35 |
| Id18 | 0,03 | 2 | 3 | 9 | Id18 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 2 | 3 | 3 | 2 | 3 | 3 | 33 |
| Id25 | 0,04 | 1 | 3 | 10 | Id25 | 2 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 3 | 3 | 33 |
| Id21 | 0,04 | 2 | 2 | 11 | Id21 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 33 |
| Id13 | 0,02 | 2 | 3 | 12 | Id13 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 1 | 3 | 3 | 29 |
| Id22 | 0,03 | 1 | 3 | 13 | Id22 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 3 | 2 | 3 | 3 | 28 |
| Id20 | 0,03 | 3 | 1 | 14 | Id20 | 0 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 28 |
| Id28 | 0,02 | 1 | 1 | 15 | Id28 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 3 | 3 | 12 |
| Id5 | 0,01 | 1 | 1 | 16 | Id5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 2 | 3 | 10 |

Рис. 2.11. Ітерація 5, Парето-оптимальне рішення id11, id4, id15

Якщо застосувати критерій $g(x)$ до інших множин Парето (табл. 2.2), отримаємо наступні множини Парето, де градації важливості врахування вразливостей є більшими (табл. 2.3).

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Id 12 | Id 14 | Id 23 | Id 27 | Id 8 | Id 29 | Id 18 | Id 21 | Id 25 | Id 13 | Id 22 | Id 20 | Id 28 | Id 5 | |
| Id12 | 0,04 | 3 | 2 | 0 | Id12 | 3 | 3 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 35 |
| Id14 | 0,04 | 3 | 2 | 1 | Id14 | 3 | 3 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 35 |
| Id23 | 0,05 | 2 | 2 | 2 | Id23 | 2 | 2 | 3 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 33 |
| Id27 | 0,05 | 2 | 2 | 3 | Id27 | 2 | 2 | 3 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 33 |
| Id8 | 0,05 | 3 | 1 | 4 | Id8 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 32 |
| Id29 | 0,03 | 3 | 2 | 5 | Id29 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 1 | 2 | 2 | 3 | 3 | 3 | 31 |
| Id18 | 0,03 | 2 | 3 | 6 | Id18 | 1 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 2 | 3 | 3 | 2 | 3 | 3 | 30 |
| Id21 | 0,04 | 2 | 2 | 7 | Id21 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 30 |
| Id25 | 0,04 | 1 | 3 | 8 | Id25 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 29 |
| Id13 | 0,02 | 2 | 3 | 9 | Id13 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 1 | 3 | 3 | 26 |
| Id22 | 0,03 | 1 | 3 | 10 | Id22 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 3 | 2 | 3 | 3 | 25 |
| Id20 | 0,03 | 3 | 1 | 11 | Id20 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 25 |
| Id28 | 0,02 | 1 | 1 | 12 | Id28 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 3 | 3 | 11 |
| Id5 | 0,01 | 1 | 1 | 13 | Id5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 2 | 3 | 9 |

Рис. 2.12. Ітерація 6, Парето-оптимальне рішення id12, id14, id23, id27, id8

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Id 18 | Id 25 | Id 29 | Id 21 | Id 22 | Id 13 | Id 20 | Id 28 | Id 5 | |
| Id18 | 0,03 | 2 | 3 | 0 | Id18 | 3 | 2 | 2 | 2 | 3 | 3 | 2 | 3 | 3 | 23 |
| Id25 | 0,04 | 1 | 3 | 1 | Id25 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 22 |
| Id29 | 0,03 | 3 | 2 | 2 | Id29 | 2 | 1 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | 21 |
| Id21 | 0,04 | 2 | 2 | 3 | Id21 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | 21 |
| Id22 | 0,03 | 1 | 3 | 4 | Id22 | 2 | 2 | 2 | 1 | 3 | 2 | 2 | 3 | 3 | 20 |
| Id13 | 0,02 | 2 | 3 | 5 | Id13 | 2 | 2 | 1 | 2 | 2 | 3 | 1 | 3 | 3 | 19 |
| Id20 | 0,03 | 3 | 1 | 6 | Id20 | 2 | 1 | 2 | 1 | 2 | 2 | 3 | 3 | 3 | 19 |
| Id28 | 0,02 | 1 | 1 | 7 | Id28 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 3 | 3 | 10 |
| Id5 | 0,01 | 1 | 1 | 8 | Id5 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 2 | 3 | 8 |

Рис. 2.13. Ітерація 7, Парето-оптимальне рішення id18, id25, id29, id21

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 | 1 | 2 | 3 | 4 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Id 22 | Id 20 | Id 13 | Id 28 | Id 5 | |
| Id22 | 0,03 | 1 | 3 | 0 | Id22 | 3 | 2 | 2 | 3 | 3 | 13 |
| Id20 | 0,03 | 3 | 1 | 1 | Id20 | 2 | 3 | 2 | 3 | 3 | 13 |
| Id13 | 0,02 | 2 | 3 | 2 | Id13 | 2 | 1 | 3 | 3 | 3 | 12 |
| Id28 | 0,02 | 1 | 1 | 3 | Id28 | 1 | 1 | 1 | 3 | 3 | 9 |
| Id5 | 0,01 | 1 | 1 | 4 | Id5 | 1 | 1 | 0 | 2 | 3 | 7 |

Рис. 2.14. Ітерація 8, Парето-оптимальне рішення id22, id20, id13

| № | Парам. 1 | Парам. 2 | Парам. 3 | | | 0 | 1 | Σ |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Id 28 | Id 5 | |
| Id28 | 0,02 | 1 | 1 | 0 | Id28 | 3 | 3 | 6 |
| Id5 | 0,01 | 1 | 1 | 1 | Id5 | 2 | 3 | 5 |

Рис. 2.15. Ітерація 9, Парето-оптимальне рішення id28

Таблиця 2.2. Розрахована множина Парето-оптимальне рішення

| Розмір збитків вразливостей | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Id 1 | Id 24 | Id 10 | Id 17 | Id 11 | Id 12 | Id 18 | Id 22 | Id 28 | Id 5 |
| Id 6 | Id 2 | Id 3 | Id 16 | Id 4 | Id 14 | Id 25 | Id 20 | | |
| | Id 9 | Id 7 | | Id 15 | Id 23 | Id 29 | Id 13 | | |
| | Id 19 | Id 26 | | | Id 27 | Id 21 | | | |
| | | | | | Id 8 | | | | |

Таблиця 2.3. Звуження початкової множини Парето на основі отриманих даних квантів інформації

| Розмір збитків вразливостей | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Id 1 | Id 6 | Id 24 | Id 19 | Id 9 | Id 2 | Id 7 | Id 10 | Id 26 | Id 16 | Id 17 | Id 15 | Id 11 | Id 23 | Id 12 | Id 25 | Id 18 | Id 22 | Id 13 | Id 28 | Id 5 |
| | | | | | | | Id 3 | | | | | Id 4 | Id 27 | Id 14 | Id 21 | Id 29 | Id 20 | | | |
| | | | | | | | | | | | | | Id 8 | | | | | | | |

### 2.2.4 Дослідження структурної ідентифікації вразливостей розроблення програмного забезпечення

Після того, як вразливості розроблення ПЗ виявлені і включені в реєстр вразливостей, виникає необхідність оцінки і визначення їх рангу окремо для кожної мети процесу/проекту (наприклад, для рамок функціональності, часу або інших ресурсів), і побудови матриці ймовірності і наслідків [8]. Ранг

вразливості дозволяє оперативно управляти реагуванням на вразливості, розташовані в різних зонах таблиці. Зони таблиці грають роль пріоритетів.

Як було вказано, на прийняття рішення про ранг по важливості врахування вразливості впливають пріоритети МПР, сформовані багато в чому на основі експертних оцінок або результатів мозкового штурму (характерно для гнучких моделей розроблення ПЗ).

Врахувуючи ці фактори, побудуємо матрицю якісної оцінки рангу вразливостей розроблення ПЗ відповідно до даних рис. 2.4 і експертних оцінок фахівців ряду відомих фірм-виробників ПЗ (*Nix Solutions LTD, Epam Systems*) [38, 39, 40].

Розглянемо приклад основних етапів розрахунку кількісної оцінки рангу вразливостей розроблення ПЗ – звуження початкової множини Парето на основі отриманих даних "квантів інформації".

Вхідні дані прикладу розрахунку:

– $m = 3$ (тривимірні вектори з критеріями важливості);

– $k = 4$ (кількість "квантів інформації", які надані експертами);

– $Id24 \succ Id9$, $Id4 \succ Id11$, $Id8 \succ Id12$, $Id25 \succ Id21$. Отримані кванти інформації за наданими експертами висновками (рис. 2.16).

– $u_1 = (0.01,-1,1) \succ 0$, $u_2 = (0.01,-1,-1) \succ 0$, $u_3 = (0.01,0,-1) \succ 0$, $u_4 = (0,-1,1) \succ 0$ розрахунок векторів відхілень, зокрема як приклад формування $u_1 = (0{,}07;4;3) - (0{,}06;5;2) = (0{,}01;-1;1)$.

Інформація про відношення переваги, задана отриманими чотирма «квантами інформації» перевірена та є несуперечливою. Застосуємо описаний алгоритм для формування нового векторного критерію. Відповідно до теореми на вхід алгоритму слід подати набір з семи векторів $\{e_1, e_2, e_3, u_1, u_2, u_3, u_4\}$, де $e_1 = (1,0,0)$, $e_2 = (0,1,0)$, $e_3 = (0,0,1)$. Таким чином довжина циклу алгоритму буде дорівнюватиме $C_7^2 = 21$.

| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| Id 1 | Id 24 | Id 10 | Id 17 | Id 11 | Id 12 | Id 18 | Id 22 | Id 28 | Id 5 |
| Id 6 | Id 2 | Id 3 | Id 16 | Id 4 | Id 14 | Id 25 | Id 20 | | |
| | Id 9 | Id 7 | | Id 15 | Id 23 | Id 29 | Id 13 | | |
| | Id 19 | Id 26 | | | Id 27 | Id 21 | | | |
| | | | | | Id 8 | | | | |
| Отримані «кванти інформації» від Scrum Master | | | | | | | | | |
| Id24>Id9 | | $u_1$ | | 0,01 | | -1 | | 1 | |
| Id4>Id11 | | $u_2$ | | 0,01 | | -1 | | -1 | |
| Id8>Id12 | | $u_3$ | | 0,01 | | 0 | | -1 | |
| Id25>Id21 | | $u_4$ | | 0 | | -1 | | 1 | |

Рис. 2.16. Кванти інформації видані експертами

Розглянемо векторні добутки всіх отриманих пар векторів, які до решти набору мають позитивні скалярні добутки. Не важко переконатися, що критерієм відбору задовольняє тільки один вектор $y_1 = \{2;0,01;0,01\}$.

В результаті знайдений один вектор $y_1$. Йому відповідає новий скалярний критерій g з компонентами $g(x) = 2x_1 + 0.01x_2 + 0.01x_3$.

Відповідно до сформульованої вище теореми, множина Парето щодо цього 1-мірного критерію буде точнішою оцінкою для множини обраних векторів варіантів у вихідній множині Парето.

Для додаткового звуження множин Парето використовуємо новий знайдений критерій $g(x)$. Нехай маємо $X = \{Id1, Id6\}$, або що те ж саме: $X = \{(0,08, 5, 3), (0,07, 4, 4)\}$. Використавши критерій $g(x)$ отримаємо $Y = \{0,08 \cdot 2 + 5 \cdot 0,01 + 3 \cdot 0,01;\ 0,07 \cdot 2 + 4 \cdot 0,01 + 4 \cdot 0,01\}$, зробивши обчислення: Y={0,24; 0,22}.

Відповідно отриманим вагам 0,24>0,22, тому Id1 ≻ Id6 – множина Парето {Id1; Id6} розділена на дві {Id1} та {Id6} з належністю по одному елементу.

Якщо застосувати критерій $g(x)$ до інших множин Парето (табл. 2.4), отримаємо наступні множини Парето, де градації важливості врахування вразливостей є більшими (табл. 2.5).

Таблиця 2.4. Розрахована множина Парето

| Сукупності множин Паретто по важливості врахування вразливості | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Id1 | Id24 | Id10 | Id17 | Id11 | Id12 | Id18 | Id22 | Id28 | Id5 |
| Id6 | Id2 | Id3 | Id16 | Id4 | Id14 | Id25 | Id20 | | |
| | Id9 | Id7 | | Id15 | Id23 | Id29 | Id13 | | |
| | Id19 | Id26 | | | Id27 | Id21 | | | |
| | | | | | Id8 | | | | |

В результаті градація по важливості врахування факторів ризику від 10 множин Парето було здійснено перехід до 21 множин Парето.

Це дозволило на 55% звузити сукупність множин Парето та більш точно обирати пріоритетність напрямків фінансування профілактичних заходів.

Як видно з табл. 2.5, основна частина організаційних, операційних, управлінських і експлуатаційних вразливостей найзбитковіші. Це говорить про важливість врахування цих вразливостей (особливо в сучасних умовах застосування гнучких методологій розроблення ПЗ).

Слід зауважити, що багато вразливостей (наприклад, *Id 18, Id 20*), на початку певної активності можуть знаходитися в зоні низького рангу, а ближче до відповідальних віх переміститися у граничні або більш критичні зони.

В той же час, ряд існуючих вразливостей незалежно від первинного рівня рангу може переміститися в більш "критичну" область (наприклад, *Id 23*, *Id 21* та ін.).

Таблиця 2.5. Звуження початкової множини Парето на основі отриманих даних "квантів інформації"

| Сукупності множин Паретто по важливості врахування вразливості | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Множина Парето | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| | *Id 1* | *Id 6* | *Id 24* | *Id 19* | *Id 9* | *Id 2* | *Id 7* | *Id 10* | *Id 26* | *Id 16* | *Id 17* | *Id 15* | *Id 11* | *Id 23* | *Id 12* | *Id 25* | *Id 18* | *Id 22* | *Id 13* | *Id 28* | *Id 5* |
| | | | | | | | | *Id 3* | | | | | *Id 4* | *Id 27* | *Id 14* | *Id 21* | *Id 29* | *Id 20* | | | |
| | | | | | | | | | | | | | | *Id 8* | | | | | | | |

Таким чином, запропонований апарат ідентифікації і якісної оцінки рангу вразливостей розроблення ПЗ дозволяє до 55% більш точно обирати пріоритетність напрямків фінансування профілактичних заходів за допомогою звуження початкової множини Парето на основі отриманих даних "квантів інформації".

### 2.2.5 Документування результатів та їх подальша пріоритизація

Наступним етапом якісного аналізу вразливості є процес документування.

Процес аналізу вразливостей слід документувати упродовж життєвого циклу всього проекту/процесу. Об'єм документування і його форма, що містить результати аналізу, залежить від конкретних цілей проведеного аналізу вразливості.

Аналіз документації відомих фірм розробників ПЗ показав, що в підсумковому документі доцільно фіксувати наступні дані:

– титульний аркуш;

– список учасників процесу якісного аналізу вразливостей розроблення ПЗ;

– анотацію;

– зміст;

– цілі і завдання проведеного якісного аналізу вразливостей розроблення ПЗ;

– опис аналізованого об'єкту;

– методологію якісного аналізу вразливостей розроблення ПЗ - початкові припущення і обмеження, що визначають межі аналізу вразливості;

– опис використовуваних методів аналізу і обгрунтування їх застосування;

– початкові дані та їх джерела;

– результати ідентифікації;

– результати якісного аналізу вразливості;

– аналіз невизначеностей результатів оцінки вразливості;

– рекомендації по роботі з вразливостями;

– висновок;

–перелік використаних джерел інформації.

Враховуючи усі запропоновані моделі та методи розроблення безпечного ПЗ КС, загальну структуру методу документування можна представити у вигляді схеми (рис. 2.17).

Таким чином, в результаті проведених досліджень на основі класифікації і структурної ідентифікації вразливостей розроблення ПЗ

удосконалено метод якісного аналізу вразливостей розроблення програмного забезпечення.

Відмінною особливістю удосконаленого методу є врахування факторів експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки ПЗ і оцінка довільного несуперечливого кінцевого набору "квантів інформації". Це дозволить до 55% звузити сукупність множин Парето та більш точно обирати пріоритетність напрямків фінансування профілактичних заходів.

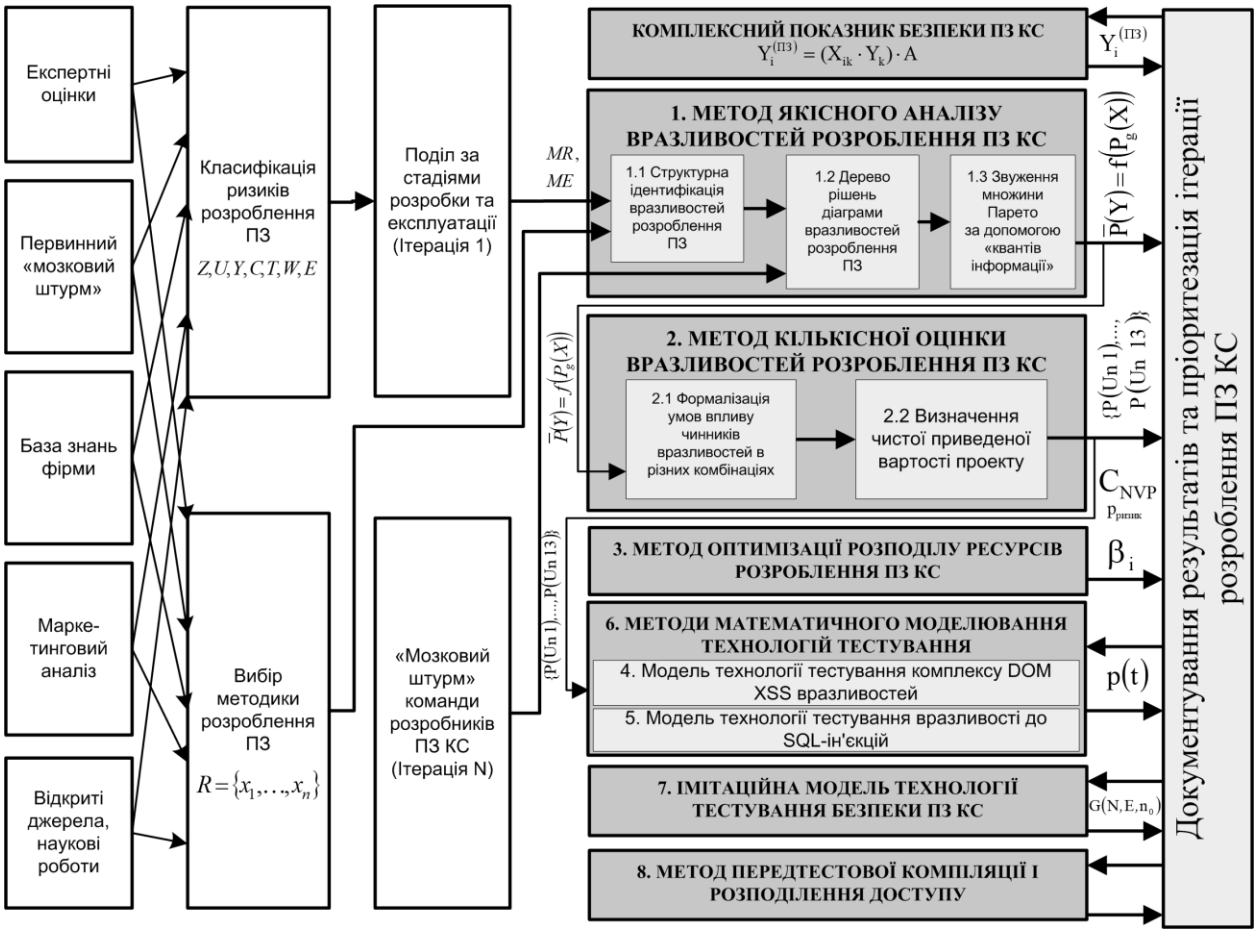


Рис. 2.17. Загальна структура методу документування ітерації

Тільки після того, як накопичено досвід і необхідний масив даних, від якісного аналізу вразливостей, доцільно переходити до їх кількісної оцінки.

## 2.3 Метод кількісної оцінки вразливостей розроблення програмного забезпечення

### 2.3.1 Використання дерева вразливостей розроблення ПЗ

Як вказано вище, для ефективного управління проектами потрібно не лише ідентифікувати вразливості, але і оцінювати їх кількісно. При цьому особливості сучасних фірм-розробників ПЗ як надсистем, особливості окремих етапів розроблення ПЗ, бізнес-процесів і їх груп як підсистем, визначають ряд проблем. До цих проблем можна віднести:

– відсутність статистичних даних про вдалі і невдалі проекти впровадження систем, особливо на операційному рівні;

– відсутність статистичних даних про провали безпеки при експлуатації ПЗ;

– унікальність кожного проекту впровадження;

– довгостроковість подібних проектів;

– високу вартість подібних проектів;

– значну складову несистемних факторів вразливості, пов'язаних з внутрішніми факторами фірми-розробника ПЗ.

Враховуючи наведені факторм можна відмітити, що для оцінки вразливостей розроблення ПЗ можна використати три основні підходи:

– формалізований опис невизначеності вразливостей розроблення ПЗ;

– коригування показників проекту шляхом заміни їх проектних значень на очікувані;

– перевірку стійкості.

Формалізована оцінка невизначеності, яка виникає в процесі реалізації проектів, за відсутності статистичних даних може спиратися на два методи: експертних оцінок і нечітких множин.

Проведені дослідження показали, що використання суб'єктивно-аксіологічної ймовірності (експертних оцінок) є вимушеним відступом науки перед нарощуванням несистемних факторів вразливості розроблення ПЗ, але це вимагає подальшої верифікації моделі і обчислених показників вразливості.

В зв'язку з цим доцільним видається перехід від суб'єктивних експертних методів до методів, які використовують теорію нечітких множин.

Коригування показників проекту (процесу розроблення ПЗ) шляхом заміни їх проектних значень на значення з урахуванням вразливостей викликає додаткові складнощі, пов'язані з невизначеністю усіх факторів, що впливають на фінансові, іміджеві і інші придбання і втрати.

Для подолання цих проблем можна використовувати методи, які спираються на опис бізнес-процесів і дозволяють виявляти зміни окремих їх параметрів, пов'язаних з розробкою, впровадженням і експлуатацією ПЗ.

Проведені дослідження показали, що адекватним інструментом для таких досліджень являється "Аналіз дерева відмов" (*Fault Tree Analysis, FTA*), запропонований в роботах [36, 41-43].

*FTA* методика описана в декількох галузевих і державних стандартах: *NUREG CPH-0492* для атомної енергетики [44]. Орієнтована на космос версія цього стандарту використовується *NASA*, стандарт *SAE ARP4761* для цивільної аерокосмічної галузі [45], *MIL-HDBK-338* - для військових систем. *IEC* стандарт призначений для міжгалузевого використання і був прийнятий як європейський стандарт *EN 61025* [46].

Аналіз цього підходу кількісної оцінки вразливостей показав доцільність використання графічної моделі *FTA* в термінах математичної логіки. Це допоможе формалізувати умови впливу факторів вразливості в різних їх комбінаціях на кінцеві показники проекту розроблення ПЗ.

Приклад дерева вразливостей розроблення ПЗ наведений на рис. 2.18.

На цій схемі групи вразливостей формуються з урахуванням розробленої класифікації вразливостей розроблення ПЗ (див. рис. 2.3) і результатів якісної оцінки рангу вразливостей (див. табл. 2.5).

Крім того, для наочності оцінки груп вразливостей пропонується використати логічні елементи "*and*" і "*or*", які дозволяють використати методи математичної логіки для розрахунку коефіцієнтів груп вразливостей і загальної вразливості.

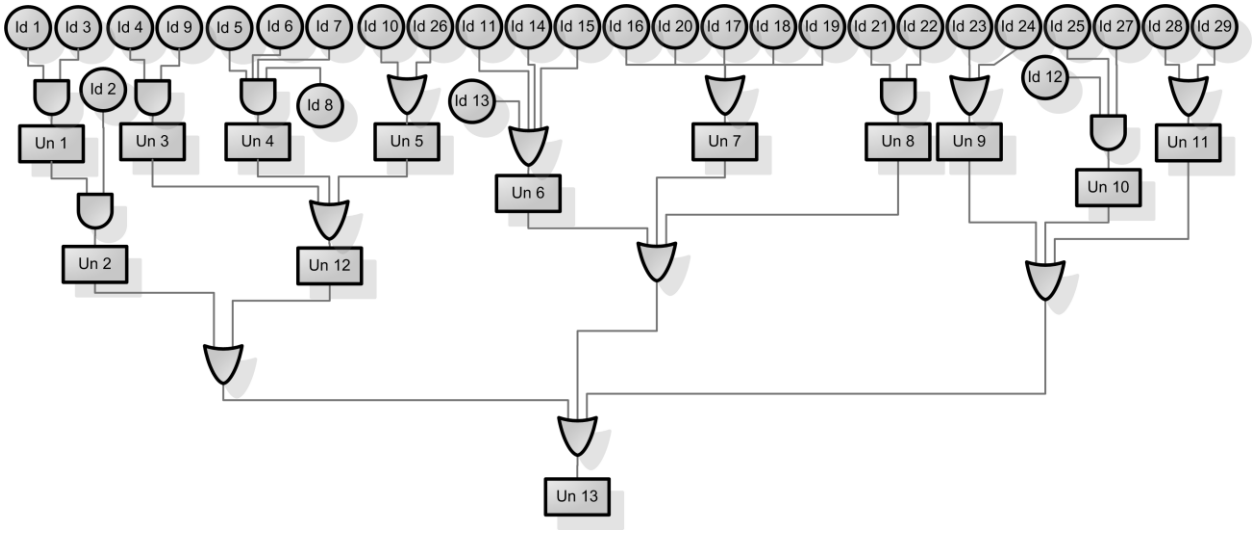


Рис. 2.18. Приклад дерева вразливостей розроблення ПЗ

Очевидно, що для цієї схеми загальний коефіцієнт вразливостей розроблення ПЗ можна розрахувати по формулі:

$$\begin{aligned} \mathrm{P}(\mathrm{Un}\,13) = 1 - ((1 - \mathrm{P}(\mathrm{Un}\,2)) \cdot (1 - \mathrm{P}(\mathrm{Un}\,12)) \cdot (1 - \mathrm{P}(\mathrm{Un}\,6)) \cdot (1 - \mathrm{P}(\mathrm{Un}\,7)) \cdot \\ \cdot (1 - \mathrm{P}(\mathrm{Un}\,8)) \cdot (1 - \mathrm{P}(\mathrm{Un}\,9)) \cdot (1 - \mathrm{P}(\mathrm{Un}\,10)) \cdot (1 - \mathrm{P}(\mathrm{Un}\,11))), \end{aligned} \quad (2.7)$$

де $P(Un\,1) = P(Id\,1) \cdot P(Id\,3)$ – коефіцієнт вразливості вибору неправильної стратегії розроблення ПЗ;

$P(Un\,2) = P(Un\,1) \cdot P(Id\,2)$ – коефіцієнт вразливості вибору неправильної методики розроблення ПЗ;

$P(Un\,3) = P(Id\,4) \cdot P(Id\,9)$ – коефіцієнт вразливості зневаги топ-менеджментом розроблення ПЗ;

$\mathrm{P}(\mathrm{Un}\,4) = \mathrm{P}(\mathrm{Id}\,5)\cdot \mathrm{P}(\mathrm{Id}\,6)\cdot \mathrm{P}(\mathrm{Id}\,7)\cdot \mathrm{P}(\mathrm{Id}\,8)$ – коефіцієнт вразливості неадекватного менеджменту активної стадії розроблення ПЗ;

$P(Un\,5) = P(Id\,10) + P(Id\,26)$ – коефіцієнт вразливості неадекватного операційного та соціального менеджменту;

$\mathrm{P}(\mathrm{Un}\,6) = 1 - ((1 - \mathrm{P}(\mathrm{Id}\,11))\cdot(1 - \mathrm{P}(\mathrm{Id}\,13))\cdot(1 - \mathrm{P}(\mathrm{Id}\,14))\cdot(1 - \mathrm{P}(\mathrm{Id}\,15)))$ – коефіцієнт вразливості невідповідності професійного рівня учасників проекту;

$P(Un\,7) = P(Id\,16) + P(Id\,17) + P(Id\,18) + P(Id\,19) + P(Id\,20)$ – коефіцієнт технологічних вразливостей розроблення ПЗ;

$P(Un\,8) = P(Id\,21)\cdot P(Id\,22)$ – коефіцієнт вразливості невідповідності складності ПЗ рівню підготовки експлуатанта;

$P(Un\,9) = P(Id\,23) + P(Id\,24)$ – коефіцієнт вразливості невідповідності експлуатації ПЗ;

$P(Un10) = P(Id\,12)\cdot P(Id\,25)\cdot P(Id\,27)$ – коефіцієнт вразливості неадекватної комунікації учасників проекту;

$P(Un11) = P(Id\,28) + P(Id\,29)$ – коефіцієнт настання правових вразливостей;

$P(Un\,12) = P(Un\,3) + P(Un\,4) + P(Un\,5)$ – коефіцієнт вразливості неадекватного менеджменту проекту.

Значення наведених вище коефіцієнтів, отриманих в результаті якісного аналізу вразливостей розроблення ПЗ КС на основі прикладу, наведені в табл. 2.6.

Використовуючи вираз (2.7) і наведені в табл. 2.5. значення коефіцієнтів можна визначити, що загальний коефіцієнт вразливостей розроблення ПЗ $P(Un13) = 0{,}541$.

Слід зауважити, що зневага – вразливістю невиявлення загроз безпеки ПЗ ($P(Id24)$) може знизити точність кількісної оцінки вразливостей

розроблення ПЗ до 13% (*P*(*Un13*)=0,488), а зневага коефіцієнтом вразливості невідповідності експлуатації ПЗ (*P(Un9)*) знизить точність оцінки до 20%.

Це підтверджує необхідність врахування вразливостей невиявлення загроз безпеки ПЗ і неадекватного взаємовпливу експлуатованого та впроваджуваного ПЗ.

Таблиця 2.6. Значення елементів дерева вразливостей розроблення ПЗ

| Коефіцієнт вразливості | Ймовірність виникнення вразливості | Коефіцієнт вразливості | Ймовірність виникнення вразливості |
|---|---|---|---|
| Un1 | 0,5% | Un8 | 0,1% |
| Un2 | 0% | Un9 | 19,9% |
| Un3 | 0,3% | Un10 | 0% |
| Un4 | 0% | Un11 | 4,9% |
| Un5 | 10,7% | Un12 | 10,9% |
| Un6 | 13,4% | **Un13** | 54,1% |
| Un7 | 21,9% | | |

### 2.3.2 Оцінка показника чистої приведеної вартості проекту розроблення безпечного ПЗ

Ще однією особливістю проектів розроблення ПЗ є їх довгостроковість і необхідність врахування інформації, що виникає на чергових стадіях прийняття рішень. Для вирішення цього завдання в [47-51], наприклад, пропонується, використовуючи модульність програмного забезпечення, оцінювати їх інвестиційну привабливість з урахуванням розроблених компонент.

Проведені дослідження дозволили знайти такий підхід ймовірнісної оцінки вразливостей, який дозволив використати основні положення нечітко-множинної теорії при оцінці ключових показників результативності проекту. Можливість його використання оцінимо на прикладі показника вартості $C_{NPV}$ *(Net Present Value)*.

Аналіз ряду робіт, пов'язаних з економічною теорією вразливостей, показав, що для розрахунку показника вартості часто використовується класичний метод визначення чистої приведеної вартості проекту:

$$C_{NVP} = \sum_{i=1}^{n} \frac{D_i}{(1+r)^i} - \sum_{i=1}^{n} \frac{C_i}{(1+r)^i}, \qquad (2.8)$$

де n – кількість періодів реалізації проекту;

$D_i$ – фінансовий потік доходів від проекту в період $i$;

$C_i$ – фінансовий потік витрат на проект в період $i$;

$r$ – ставка дисконтування.

Однак, з урахуванням того, що проект розроблення безпечного ПЗ є довгостроковим і залежить від багатьох факторів, існує необхідність розглядати показники повернення інвестицій в ці проекти спільно з процесами розвитку проекту. Окрім цього, важливо враховувати фактори динамічної зміни небезпечних компонентів в процесі проектування, кодування, тестування і експлуатації на всіх "витках спіралі" розроблення безпечного ПЗ.

Проведені дослідження показали, що з урахуванням додаткових факторів вразливості, а також (2.8) для розрахунку $C_{NPV}$ доцільно використати наступний вираз:

$$\mathrm{C_{NVP}} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{(\mathrm{B} - \mathrm{AC_i})}{(1+\mathrm{r})^{\mathrm{i}}} - \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{\left(\sum_{\mathrm{k=1}}^{\ell} \mathrm{C_i^{(k)}}\right)}{(1+\mathrm{r})^{\mathrm{i}}} - \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{\left(\sum_{\mathrm{k=1}}^{\ell} \mathrm{e}^{-\lambda \times \mathrm{QC_i^{(k)}}}\right)}{(1+\mathrm{r})^{\mathrm{i}}}, \qquad (2.9)$$

де $B$ - фінансові інвестиції, що поступають в процесі розроблення ПЗ в період $i$;

$AC_i$ - поточні витрати на підтримку і розвиток системи розроблення в період $i$;

$\ell$ - кількість додаткових видів витрат на придбання, налаштування і модернізацію технічної, технологічної, програмної та інших складових в процесі розроблення ПЗ;

$C_i^{(k)}$ - витрати на врахування безпеки та тестування вразливостей ПЗ;

$\lambda$ - параметр впливу факторів безпеки та тестування вразливостей ПЗ на чисту приведену вартість проекту.

Відмінною особливістю математичного виразу (2.9) є введення додаткових складових $\left( \sum_{i=1}^{n} \frac{\left( \sum_{k=1}^{\ell} C_i^{(k)} \right)}{(1+r)^i}; \sum_{i=1}^{n} \frac{\left( \sum_{k=1}^{\ell} e^{-\lambda \times C_i^{(k)}} \right)}{(1+r)^i} \right)$, які характеризують врахування додаткових витрат на апаратну та програмну модернізацію фірми, вдосконалення системи її управління, а також врахування безпеки та тестування вразливостей ПЗ.

Оцінка ефективності проекту полягає у порівнянні $C_{NVP}$ з деяким значенням $C_{NVP\text{reference}}$, яке визначає мінімально допустимий рівень приведеної вартості проекту розроблення безпечного ПЗ.

З урахуванням невизначеності складових, на основі теорії нечітких множин можна розглядати трикутні нечіткі значення складових для вираження (2.9): $\overline{\overline{B_i}} = \left( B_i^{(\min)}, \overline{B_i}, B_i^{(\max)} \right)$ якщо неможливо визначити фінансові доходи, які виникнуть від модифікації в процесі розроблення ПЗ технологічних етапів; $\overline{\overline{AC_i}} = \left( AC_i^{(\min)}, \overline{AC_i}, AC_i^{(\max)} \right)$ якщо існує невизначеність витрат, необхідних для підтримки і розвитку системи розроблення безпечного ПЗ; $\overline{\overline{C_i^{(k)}}} = \left( C_i^{(k)(\min)}, \overline{C_i^{(k)}}, C_i^{(k)(\max)} \right)$ якщо існує невизначеність відносно додаткових витрат *k*-го призначення.

Крім того, подібним же чином треба представити коефіцієнт дисконтування: $\overline{\overline{r_i}} = \left(r_i^{(\min)}, \overline{r_i}, r_i^{(\max)}\right)$ якщо інвестор не може оцінити вартість капіталу, який буде використаний в проекті.

Для приведення формули (2.9) до вигляду, який може використовуватися для обчислень, скористаємося сегментним способом.

Якщо обрати фіксований рівень приналежності нечітких чисел (ординату функції приналежності нечітких чисел), то можна застосувати операції нечіткої арифметики, які дозволять перетворити вираз (2.9) на систему рівнянь:

$$\left[C_{NVP1}, C_{NVP2}\right] = \begin{cases} \sum_{i=1}^{n} \frac{\left(B_{i_1} - AC_{i_1}\right)}{(1+r)^i} - \sum_{i=1}^{n} \frac{\left(\sum_{k=1}^{\ell} C_{i_1}^{(k)}\right)}{(1+r)^i} - \sum_{i=1}^{n} \frac{\left(\sum_{k=1}^{\ell} e^{-\lambda \times C_{i1}^{(k)}}\right)}{(1+r)^i}, \\ \sum_{i=1}^{n} \frac{\left(B_{i_2} - AC_{i_2}\right)}{(1+r)^i} - \sum_{i=1}^{n} \frac{\left(\sum_{k=1}^{\ell} C_{i_2}^{(k)}\right)}{(1+r)^i} - \sum_{i=1}^{n} \frac{\left(\sum_{k=1}^{\ell} e^{-\lambda \times C_{i2}^{(k)}}\right)}{(1+r)^i}. \end{cases} \quad (2.10)$$

Дослідимо і проведемо оцінку вразливостей розроблення ПЗ за наступних умов: $n$=3; $B_1$=15; $B_2$=16; $B_3$=17; $r$ = (0,1; 0,125; 0,15); $AC_1$=$AC_2$=$AC_3$=1; $\ell = 3$ (1 - витрати на придбання, налаштування і модернізацію програмного забезпечення в ході реалізації проекту, 2 - витрати на модернізацію апаратного забезпечення фірми і реорганізацію системи управління, яка потрібна після модернізації програмного забезпечення і устаткування, 3 - витрати на врахування безпеки ПЗ і додаткове тестування його вразливості); $C_{i1} = (1{,}1; 1{,}2; 1{,}3)$; $C_{i2} = (0{,}5; 0{,}6; 0{,}7)$; $C_{i3} = (0{,}1; 0{,}2; 0{,}3)$, $\lambda = 1$, $C_{NVP\text{reference}} = 0$.

На рис. 2.19 представлені графіки залежності $\left[C_{NVP1}, C_{NVP2}\right]$ від значення $B$ фінансових інвестицій, що поступають в процесі розроблення безпечного ПЗ в період $i$ (графіки CN3 і CN4 ілюструють залежність за відсутності додаткових витрат на врахування безпеки і тестування вразливості ПЗ).

Як видно з цього рисунка, введення додаткових витрат в процесі розроблення безпечного ПЗ до 1,5 разів підвищує чисту приведену вартість проекту.

На рис. 2.20 показано залежність вартості проекту в залежності від вкладених коштів в засоби запобіганню вразливостям.

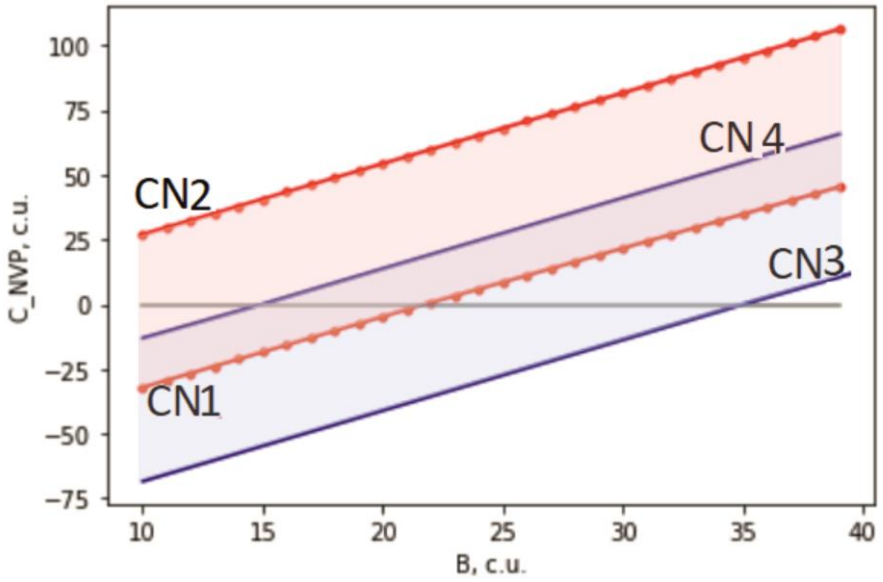


Рис. 2.19. Графіки залежностей $[C_{NVP1}, C_{NVP2}]$ від значення $B$

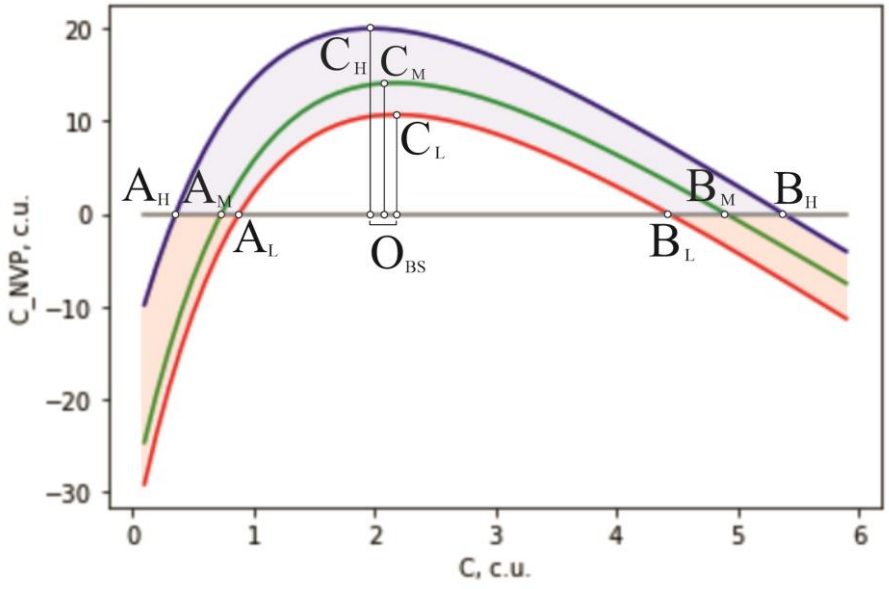


Рис. 2.20. Графік залежності $C_{NVP}$ від значення $C$

Очевидно, при фінансуванні заходів уникнення вразливостей до певного моменту збільшують вартість проекту за рахунок зменшення витрат на вирішення проблем, які є результатом вразливостей, які здійснилися.

Однак, згодом ймовірність виникнення вразливих ситуацій зменшується не так значно, тому фінансові вливання починають переважати отримані прибутки, це виражається в поступовому переході графіку знову до від'ємних величин. В результаті маємо наявну оптимальну точку фінансування профілактичних заходів, при якій вартість проекту є максимальною. Для максимального значення, мінімального та моди нечіткого значення ця точка оптимуму відрізняється на незначну величину (сама точка оптимуму стає нечітким числом).

Фінансування запобіжних заходів є ситуативною подією, яка залежить від поточного фінансового стану, важливості проекту для виконавця та замовника, тому в більшості можна вважати суму фінансування, як число, яке лежить, наприклад, на проміжку [0, 6] у.о.

Саме така ситуація і показана на рис. 2.9. У результаті можна оцінити ступінь ризику як ймовірність отримання від'ємної вартості проекту. Для цього потрібно знайти відношення від'ємної частини площі графіку до загальної площі фігури між графіком та кривою:

$$p_{вр} = \frac{\int_{C_{NVP}<0} C_{NVP}(C)dC}{\int_{0}^{6} |C_{NVP}(C)|dC} \qquad (2.11)$$

Для наведеного прикладу проекту розроблення ПЗ, за результатом чисельного інтегрування маємо нечітку ймовірність рівня вразливості що вартість ітерації (спринту) буде від'ємною: $p_{вр} = (0{,}034; 0{,}219; 0{,}421)$.

Таким чином, запропонований в підрозділі 2.3.2 загальний спосіб оцінки показника чистої приведеної вартості проекту розроблення

безпечного ПЗ є засобом для подолання проблем, пов'язаних з неефективним завершенням проектів розроблення ПЗ.

Спосіб пропонує розглядати проект комплексно, з урахуванням необхідності врахування безпеки і тестування вразливості ПЗ, з використанням інструментів, які дозволяють здолати складність, невизначеність і довгостроковість проектів.

Для подолання проблем відсутності статистичних даних і підвищення достовірності експертної оцінки пропонується використати методи теорії нечітких множин.

Враховуючи всі описані вище етапи кількісної оцінки вразливостей розроблення ПЗ можна відмітити, що відмінною особливістю розробленого методу є комплексне використання "Аналізу дерева відмов" і способу оцінки показника чистої приведеної вартості проекту розроблення безпечного ПЗ з урахуванням негативних факторів можливого невиявлення загроз безпеки ПЗ.

Результати кількісної оцінки вразливостей розроблення ПЗ можуть бути використані в якості вхідних даних при управлінні вразливостями безпосередньо на подальших етапах розроблення безпечного ПЗ, представлених на рис. 2.2. (прототипування, кодування, тестування і т.і.).

## 2.4 Висновки до розділу

У розділі визначено і вирішено одне з протиріч, що виникають при розробці безпечного ПЗ, яке полягає в зневазі фірмами-розробниками ПЗ факторів уразливості безпеки ПЗ.

В якості вирішення вказаної проблеми запропоновано використання розроблених методів якісного аналізу і кількісної оцінки вразливостей розроблення програмного забезпечення.

В ході рішення поставленої задачі на першому етапі розроблено метод якісного аналізу вразливостей розроблення програмного забезпечення.

Його відмінною особливістю є врахування факторів експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки ПЗ і оцінкою довільного несуперечливого кінцевого набору "квантів інформації". Це дозволить до 55% звузити сукупність множин Парето та більш точно обирати пріоритетність напрямків фінансування профілактичних заходів.

Однією з основних складових методу є структурна ідентифікація вразливостей розроблення ПЗ, що відрізняється від відомих побудовою оцінки вразливостей розроблення ПЗ "зверху" у вигляді множини, за наявності довільного несуперечливого кінцевого набору "квантів інформації".

На другому етапі розроблено метод кількісної оцінки вразливостей розроблення ПЗ. Його відмінною особливістю є комплексне використання "Аналізу дерева відмов" і способу оцінки показника чистої приведеної вартості проекту розроблення безпечного ПЗ з урахуванням негативних факторів можливого невиявлення загроз безпеки ПЗ.

Використання вдосконаленого "Аналізу дерева відмов" дозволить до 20% підвищити точність кількісної оцінки вразливостей розроблення ПЗ. В той же час, використання способу оцінки показника чистої приведеної вартості проекту розроблення безпечного ПЗ дозволить розглядати проект комплексно, з урахуванням необхідності врахування безпеки і тестування вразливості ПЗ, із залученням інструментів, які дозволяють здолати складність, невизначеність і довгостроковість проектів.

Отпубліковані наукові результати роботи автора приведені у 2 розділі [38, 39, 40, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64].

**Список використаних джерел:**


[1] Gavrylenko S. Software security overview /Anoushirvan Rashidinia, S. Gavrilenko, M. Pochebut, O. Sytnikova // Системи управління навігації та зв'язку. – ПНУ, 2019. – Вип. 2 (54) с.55-58.

[2] Sherman M. Building Secure Software for Mission Critical Systems [Електронний ресурс] – Режим доступу: http://resources.sei.cmu.edu/asset_files/Presentation/ 2017_017_001_495865.pdf

[3] Благодатских В.А. Стандартизация разработки программных средств / В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов – М: Финансы и статистика, 2003. – 288 с.

[4] Макконнелл С. Совершенный код. Мастер-класс. М.: Рус. Редакция ; СПб.: Питер, 2007. 896 с.

[5] Allan, H. Visual Specification of Security / H. Allan, M. W. Maimone, J. D. Tygar, J. M. Wing, M. Zaremski, A. Miry. // IEEE Transactions on Software Engineering (TSE), 1990. – 16(1). – P. 1185-1197.

[6] Seacord Robert The CERT® C Coding Standard, Second Edition: 98 Rules for Developing Safe, Reliable, and Secure / Robert C. Seacord – Addison-Wesley, 2014. - pp 576.

[7] Seacord Robert The CERT C Secure Coding Standard / Robert C. Seacord – Addison-Wesley, 2008. - pp 720.

[8] Семенов С.Г., Лисица Д.А. Модель оценки риска разработки программного обеспечения: сб. материалов XV Международной НТК «Проблемы информатики и моделирования». Харьков: НТУ «ХПИ», 2015. С.82.

[9] Бриткин А. И. Риски, связанные с внедрением технологий, в проектах

разработки программного обеспечения / А. Бриткин // Социально-экономические и технические системы. – 2007. – № 8 (42).

[10] Вишняков Я.Д Общая теория рисков: учеб. пособие для студ. высш. учеб. заведений / Я.Д.Вишняков, Н.Н.Радаев. – 2-е изд., испр. - М. : Издательский центр «Академия», 2008. – 368 с.

[11] Шапкин А.С. Теория риска и моделирование рисковых ситуаций / А.С. Шапкин, В.А. Шапкин.– М.: Издательско-торговая корпорация «Дашкв и К», 2005. – 880 с.

[12] Stoneburner Gary Goguen, Alice, and Feringa Alexis Risk Management Guide for Information Technology Systems Recommendations of the National Institute of Standards and Technology // Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, 2002. - 55 p.

[13] Вендров А.М. Проектирование программного обеспечения экономических информационных систем / А.М. Вендров. – М.: Финансы и статистика, 2005. – 544 с.

[14] Гусятников В.Н. Стандартизация и разработка программных систем / В.Н. Гусятников, А.И. Безруков – М: Финансы и статистика, 2010. – 288 с.

[15] Seacord Robert Secure Coding in C and C++ / Robert C. Seacord – Addison-Wesley, 2013. – 600 p.

[16] Zhang Peng. Security in Network Coding / Peng Zhang, Chuang Lin. – Springer, 2016. – 98 p.

[17] Zeng Y. Risk Management For Enterprise Resource Planning System Implementations in Project-Based / Y. Zeng // Firms : dis. for the degree of PHD, Maryland, 2010. – P. 210.

[18] Вэйдер Майкл Томас. Инструменты бережливого производства.

Мини-руководство по внедрению методик бережливого производства / Майкл Томас Вэйдер – Альпина Паблишер. 2012. – 125 с.

[19] Hasan Yasar Security Practitioner Perspective on DevOps for Building Secure Solutions / Режим доступу: http://www.sei.cmu.edu/webinars/view_webinar.cfm?webinarid= 474101 &ga Webinar= Security Practitioner Perspectiveon DevOpsforBuilding Secure Solutions

[20] OSSTMM 3. Open-Source Security Testing Methodology Manual [Електронний ресурс] – Режим доступу: http://www.isecom.org/mirror/OSSTMM.3.pdf

[21] Patcha, A. An overview of anomaly detection techniques: Existing solutions and latest technological trends / A. Patcha, J. M. Park : Computer Networks. – 2007. – Vol. 51, №. 12. – pp. 3448-3470.

[22] Seacord Robert Modernizing Legacy Systems / Robert C. Seacord, Daniel Plakosh, Grace A. Lewis – Addison-Wesley, 2003. –332 p.

[23] Swiderski Frank Threat Modeling / Frank Swiderski, Window Snyder. – Microsoft Press, 2004. 257 p.

[24] Test & Test Case Management in Jira [Електроний ресурс] – Режим доступу: http://blog.alsedi.com/test-test-case-management-in-jira-part-0/

[25] Когда «Agile» (не) к месту [Електронний ресурс]. – Режим доступу до ресурсу: https://makhmetov.ru/articles/agile.html

[26] Kniberg Henrik Scrum and XP from the Trenches - 2nd Edition / Henrik Kniberg – InfoQ, 2015. – 94p.

[27] Putu Adi Guna Permana Scrum Method Implementation in a Software Development Project Management/ Putu Adi Guna Permana // International Journal of Advanced Computer Science and Applications.

– 2015. – Vol. 6. № 9. – P. 199-205.

[28] Soumya Krishnan M. Software Development Risk Aspects and Success Frequency on Spiral and Agile Model. / M. Soumya Krishnan // International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 1, January 2015 pp.301-310.

[29] Boehm B.W. A spiral model of software development and enhancement / Boehm B., Egyed A. // IEEE Computer, May 1988 pp. 61-72.

[30] Boehm Barry W. Balancing Agility and Discipline - A Guide for the Perplexed / Barry W. Boehm, Richard Turner // Addison-Wesley. – 2004. – P. 605-609

[31] Demarco T., Lister T. Waltzing with bears: Managing risk on software projects. New York, 2003. 342 p.

[32] Исикава К. Японские методы управления качеством / К. Исикава, Сокр.пер. с англ. / Под. Ред. А. В. Гличева. – М: Экономика, 1988. – 214 с.

[33] Гуру мендеджмента качества и их концепций: Э.Деммин, Дж.Джуран, Ф.Кросби, К.Исикава, А.Фейгенбаум, Т.Тагути / [Електронний ресурс]. – Режим доступу: http:// www.management. com.ua /qm /qm009.html

[34] Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

[35] Аоки М. Оптимизация стохастических систем / М. Аоки. – М.: «Наука» ФИЗМАТЛИТ, 1971. – 424 с.

[36] Gavrylenko S. Development of anomalous computer behavior detection method based on probabilistic automaton. Monograph/ Gavrylenko S., Chelak V., Semenov S., E. Chelak/ Kiev, 2019. – c. 237-258

[37] Шибанов, А.П. Обобщенные GERT-сети для моделирования протоколов, алгоритмов и программ телекоммуникационных систем: дис. доктора техн. наук: 05.13.13 [Текст] / Шибанов Александр Петрович. – Рязань, 2003. – 307 с.

[38] Коваленко А.В. Метод качественного анализа рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

[39] Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розробки програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

[40] Коваленко А.В. Методы качественного анализа рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016). м. Харків. 30 березня – 1 квітня 2016 р. – Харків: НТУ «ХПІ». – 2016. – С. 6-7.

[41] Geymayr J. Fault-Tree Analysis: A Knowledge-Engineering Approach / J. Geymayr, N. Ebecken // IEEE Transactions on Reliability. – 1995. – № 44(1), pp. 37 – 45.

[42] Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты. Киев, 2002. 688 с.

[43] Heard N.A. Bayesian anomaly detection methods for social networks / N.A. Heard, D.J. Weston, K. Platanioti, D.J. Hand : Ann. Appl. Stat. – 2014.– vol. 2.– P. 645 – 662.

[44] Vesely W.E. Fault Tree Handbook (NUREG-0492) Режим доступу: https://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/

[45] Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment ARP4761 [Електронний ресурс]– Режим доступу: https://www.sae.org/standards/content/arp4761/

[46] BS EN 61025:2007 [Електронний ресурс]. – Режим доступу: https://www.en-standard.eu/bs-en-61025-2007-fault-tree-analysis-fta/?gclid=EAIaIQobChMIyLCnroGp5wIVCM-yCh2CDgQfEAAYASAAEgJKH_D_BwE

[47] McConnell S. Software Estimation: Demystifying the Black Art (Developer Best Practices). Microsoft Press, 2006. 308 p.

[48] Executive summary agility accelerates the delivery of business value [Електронний ресурс] – Режим доступу: http://www.agile247.pl/wp-content/uploads/2017/04/versionone-11th-annual-state-of-agile-report.pdf

[49] Combee Janine Discussion paper Soft controls What are the starting points for the internal auditor? / Janine Combee, Mandy Dijkman, The institute of Internal auditors Niderlands, 2015 [Електронний ресурс] – Режим доступу: https://www.iia.nl/SiteFiles/Publicaties/ IIA_Bro %20 A4 %20Soft %20Controls %20Engels%2002.pdf

[50] Мораско Дж. IT-проекты: фронтовые очерки. СПб.: Символ-Плюс, 2007. 384 с.

[51] Denning P.J. Discussing Cyber Attack / P.J. Denning, D.E. Denning // Comm. of the ACM. – 2010. – Vol. 53. – №. 9. – P. 29-31.

[52] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Информационные технологии в управлении, образовании, науке и промышленности:

монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

[53] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 5(142). – Х.: ХУПС – 2016. – С. 153-157.

[54] Коваленко А.В. Метод количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

[55] Коваленко А.В. Анализ и оценка рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез «Securitea informationala 2015-2016». Conferenta internationala (editia a XII-a). Chisinau. Moldova. 3 martie 2016. – Chisinau: ADSEM. – 2016. – P. 96-102.

[56] Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

[57] Коваленко А.В. Исследование дерева рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез III міжнародної науково-практичної

конференції «Інформаційна та економічна безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХНМІ ДВНЗ «УБС». – 2016. – С. 174-178.

[58] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Сборник тезисов XII международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 30 мая – 02 июня 2016 г – Варна. ТУВ. – 2016. – С. 585-589.

[59] Kovalenko Oleksandr Qualitative risk analysis of software development / Oleksandr Kovalenko, Jamil Al-Azzeh, Oleksii Smirnov, Anna Kovalenko, Serhii Smirnov // Asian Journal of Information Technology. – Volume 17 Issue 3. – Medwell Journals. – 2018. – P. 218-230. ISSN: 1682-3915. URL: http://medwelljournals.com/ abstract/ ?doi= ajit.2018. 218. 230 Doi: ajit.2018.218.230

[60] Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". – Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

[61] Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розроблення програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

[62] Коваленко А.В. Методика структурной идентификации рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології»

(IS&CT). м. Кіровоград. 24-25 березня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 71-72.

[63] Коваленко А.В. Структурная идентификация рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 15-16 квітня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 175-182.

[64] Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Conferenta internationala (editia XIII-a). «Securitea informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

# РОЗДІЛ 3
# МЕТОД ОПТИМІЗАЦІЇ РОЗПОДІЛУ РЕСУРСІВ РОЗРОБЛЕННЯ БЕЗПЕЧНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В розділі розглянуто задача оптимізації розподілу ресурсів розроблення ПЗ за умови обмеженості ресурсів (фінансових, технічних та ін.), які маються у наявності на усунення помилок безпеки. Процес розглядається як напівмарківська модель прийняття рішень для керованого процесу у безперервному часі з критерієм мінімуму витрат на усунення аномалій.

Удосконалено метод оптимізації розподілу ресурсів розроблення ПЗ, що відрізняється від відомих використанням псевдобулевих методів бівалентного програмування з нелінійною цільовою функцією і лінійними обмеженнями для визначення оптимальної стратегії усунення експлуатаційних помилок.

У якості прикладу розглянуто ситуації виникнення помилок безпеки ПЗ і визначено оптимальну стратегію управління для усунення вказаної аномальної ситуації.

## 3.1 Постановка завдання

Проведені дослідження, а також аналіз літератури [1-7] показали, що забезпечення якості ПЗ полягає у площині завчасного виявлення пов'язаних з вразливостю фінансових, технічних, психологічних, та інших вразливостей, і проведеня заходів по зниженню вразливості шляхом цілеспрямованої зміни цих факторів з урахуванням ефективності заходів, що приймаються. Управління вразливостями розроблення ПЗ включає систему заходів, здійснюваних як до прояву негативної події, так і після її реалізації. Проте, як показали дослідження, превентивний аналіз і врахування більшості

можливих експлуатаційних помилок дозволить понизити фінансові та інші витрати в життєвому циклі розроблення ПЗ.

Ряд авторів [1, 2, 8-12] під терміном "управління вразливостю" розуміють розроблення і обґрунтування оптимальних програм діяльності, закликаних ефективно реалізувати рішення в області забезпечення безпеки. При цьому головним елементом такої діяльності є процес оптимального розподілу обмежених ресурсів з урахуванням характерних експлуатаційних, економічних і соціальних факторів.

Процес оптимального розподілу ресурсів розроблення ПЗ при певних обмеженнях на заходи по тестуванню якості і безпеки, формалізуємо як напівмарківську модель оптимізації керованого марківського процесу з врахуванням отриманих дивидентів з коефіцієнтом $0 < \alpha < 1$ при нормальних умовах процесу створення ПЗ, або витратами (в умовах з відхиленнями від плану, пов'язаними зі зневагою невиявлення вразливостей (помилок) безпеки). При цьому цей вид експлуатаційних вразливостей ототожнюється з множиною послідовно сполучених незалежних станів, які можливо описати, які мають часові обмеження.

Стратегію оптимізації розподілу ресурсів розроблення ПЗ визначимо з використанням псевдобулевих методів бівалентного програмування, з врахуванням вищепозначених обмежень. Для знаходження базисних рішень системи лінійних нерівностей з булевими змінними використаємо відомий та перевірений у [13] алгоритм перетину рішень окремих нерівностей-обмежень.

У таких умовах класичний, визначений у роботах [13, 14], опис постановки завдання трактується наступним чином.

Нехай кожному стану $i \in S$, де $S = \{0,1,2,..,N\}$ системи розроблення ПЗ поставлена у відповідність скінченна множина $R_i$ рішень оптимального розподілу ресурсів, елементи якої позначимо як $r = 1,2,...,r_i$. Якщо система

знаходиться в стані $i \in S$ і приймається рішення $r = R_i$, то її подальша поведінка визначається ймовірнісним законом

$$Q_{ij}^{r}(t) = P_{ij}^{(r)} F_{ij}^{(r)}(t), \quad j \in S \tag{3.1}$$

де $P_{ij}^{(r)}$ – ймовірність переходу системи зі стану *i* в стан *j*;

$F_{ij}^{(r)}(t)$ – функція розподілу часу перебування системи в стані *i* при прийнятті рішення *r* і за умови, що наступний перехід відбудеться в стан *j*.

При цьому зробимо припущення:

Стан $i = 0$ відповідає нормальному процесу розроблення ПЗ, а $i \neq 0$ - ситуація помилки безпеки.

Функції $F_{0j}^{(r)}(t)$ і $F_{j0}^{(r)}(t), \quad j \in \widetilde{S} = S / \{0\}, \quad r \in R_j$, разом зі своїми першими похідними безперервні при $t > 0$, за винятком мережних точок, і відповідають математичній формалізації експоненційного закону розподілу.

За час перебування в стані $i$ у разі прийняття рішення про розподіл ресурсів $r$ використовується в середньому $k_i^{(r)}$ ресурсів (при $i \neq 0$ число $k_i^{(r)}$ негативне і дорівнює витратам системи за час перебування в стані $i$ за умови виходу з цього стану з урахуванням розподілу $r$).

Значення показників $k_i^{(r)}$ обмежені при всіх $i \in S$, $r \in R_i$ і ймовірності $P_i^{(r)}$ задовольняють співвідношенням:

$$i \in S, \quad r \in R_i, \; P_{ij}^{(r)} \geq 0, \; i, j \in S, \quad r \in R_i.$$

Таким чином, в кожному стані $i \in S$ існує $r_i$ варіантів розподілу ресурсів для розроблення ПЗ з множини $R_i$. Вибір деякого варіанту $r$ з $R_i$ у стані $i \in S$ означає задання величин $Y_{ij}^{r}(t), \; P_{ij}^{(r)}, \; F_{ij}^{(r)}(t), \; k_i^{(r)}, \; j \in S$.

При $i = 0$, $R_0 = \{0\}$, ймовірність $P_{0j}^{(r)} \neq 0, \quad j \in S$ є ймовірністю переходу у стан $j$. Ймовірність $P_{0j}^{(r)} \neq 0, \quad j \in S$ розраховується як доля станів з помилками безпеки типу *j* в множині вразливостей безпеки різних типів на

основі вхідних даних для розроблення ПЗ. В цьому випадку $F_{0j}^{(r)}(t)$ - функція розподілу часу тестової експлуатації ПЗ між виявленими помилками безпеки типу $j$.

При $i = 1,...,N$ для будь-якого $r \in R_i$, $P_{i0}^{(r)} = 1$, $P_{ij}^{(r)} = 0$, $j \neq 0$, функція $F_{i0}^{(r)}(t)$ - це функція розподілу часу усунення вразливостей безпеки на основі даних про $r$ при врахуванні помилки типу $j$.

Якщо в стані $j$ збитки $Z_j$ в одиницю часу, витрати на заходи по організації прийнятих рішень $B_j^{(r)}$ (теж на одиницю часу). То разові витрати по рішенню $r$ є $B_j^{(r)}$ на одиницю часу складають:

$$M_{ij} = t(Z_i + B_j^{(r)})P_{ij}$$

Повні витрати в середньому є:

$$M_i = \sum_j t(Z_i + B_j^{(r)})P_{ij}$$

Сумарні витрати:

$$M_i = \sum_j (Z_i + B_j^{(r)}) \int_0^t Q_{ij}^{(r)}(\tau) \cdot d\tau;$$ або для стаціонарної стратегії

$$M_i = t\sum_j (Z_i + B_j^{(r)}) \cdot Q_{ij}^{(r)}(t), t \to \infty$$

$$M = \sum_i M_i \qquad (3.2)$$

Задача мінімізувати витрати $M \to \min$.

$Q_{ij}^{(r)}(t)$ – невідомі функції ймовірності в умовах обрання множини рішень $r$.

$P_{ik}^{(r)}$ – невідомі стіціонарні ймовірності в умовах обрання множини рішень $r$.

Аналіз літератури показав, що найбільш популярна інформація про напівмарківські процеси і керовані напівмарківські моделі з додатковими витратами і дивідендами викладена в роботах [16-19].

Нехай $g_i(t,\beta)$ сумарна витрата системи, відповідно до стратегії $\beta$, за час $t$ життєвого циклу розроблення ПЗ. Обов'язково слід зазначити початок процесу при $t=0$ зі стану $i$. Також нехай $\upsilon_i(t,\beta)=g_i(t,\beta)/t$ - сумарна середня витрата фірми за час $t$ за тих же умов.

Зазначимо що $c_{rj}$ - витрати, для реалізації плану дій $r$ визначеної стратегії у разі події порушення безпеки ПЗ $j$ і $x_{rj}$ - булева змінна: $x_{rj}=1$, якщо $r$ застосовується при події $j$, $x_{rj}=0$ якщо застосування $r$ не потрібно.

Нехай загальна кількість ресурсів, відпущених для усунення недоліків безпеки ПЗ (заходи типу $r$) обмежена деякою мірою $b_r$.

Якщо витрати $c_{rj}$ задовольняють обмеженням, то реалізована організація визначає у просторі $\Re^d$, $d=\dim R$ деяку скінченну множину дискретних точок.

Тоді, у відповідності до робіт [13, 20], існує стаціонарна стратегія $\beta^*$ з відсутністю ознак псевдовипадковості, яку можно визначити як $\beta$-оптимальну.

Вона мінімізує сумарні витрати $\upsilon(\beta)$ при використання деякої стратегії керованих дій $\beta$. При цьому $\upsilon(\beta)$ є $(N+1)\times 1$ - мірний вектор $(\upsilon_0(\beta),\upsilon_1(\beta),\ldots,\upsilon_N(\beta))$, де

$$\upsilon_i(\beta)=\lim_{t\to\infty}\upsilon_i(t,\beta),\ i\in S. \tag{3.3}$$

Необхідно знайти оптимальну нерандомізовану марківську стратегію $\beta^*$, без ознак псевдовипадковості, що мінімізує сумарну середню витрату $\upsilon(\beta)$ при умові

$$y=(y_0,\ y_1,\ \ldots,\ y_N) \tag{3.4}$$

$$\sum_{i\in S} y=1,\ y\geq 0,\ i\in S \tag{3.5}$$

Проведені дослідження показали, що поставлене вище завдання можливо вирішити за допомогою бівалентного програмування з використанням псевдобулевих методів.

При цьому як вхідні дані зазначимо вектор $y = (1, 0, \ldots, 0)$ як початковий стан організації.

## 3.2 Розроблення оптимізаційної стратегії напівмарківської моделі розподілу ресурсів проектування безпечного ПЗ

Проведені дослідження показали, що ймовірності переходів розглянутого для системи розроблення ПЗ напівмарківського процесу розподілу ресурсів в моменти переходів з $i$ в $j$ для визначених рішень $r \in R_i$ описуються стохастичною $(N+1)\times(N+1)$ матрицею $P^{(r)} = \left\{p_{ij}^{(r)}\right\}$, яка характеризує вкладений ланцюг Маркова.

Елементи $p_{ij}^{(r)}$ при будь-яких $i, j \in S$ і $r \in R_i$ згідно (3.1) допомагають визначити ймовірність $p_{ij}^{(r)}$ того, що час перебування в стані $i$ не перевершує $t$.

Можливий перехід із стану $i$ в стан $j$ зі ймовірністю $p_{ij}^{(r)}$ при $r \in R_i$. Функції $Q_{ij}^{(r)}(t)$ в (3.1) задовольняють умовам

$$Q_{ij}^{(r)}(0) = 0, \ \ i, j \in S, r \in R_i, \tag{3.6}$$

$$\sum_{j \in S} Q_{ij}^{(r)}(\infty) = \sum_{j \in S} p_{ij}^{(r)} = 1, \ \ i, \in S, r \in R_i \tag{3.7}$$

Визначимо функцію

$$H_i^{(r)}(t) = \sum_{j \in S} Q_{ij}^{(r)}(t), \quad i, \in S, r \in R_i, \tag{3.8}$$

як функцією розподілу часу перебування системи у стані $i$ при прийнятті рішення про розподіл ресурсів $r \in R_i$.

Випадковий процес $(Z_t), t \geq 0$ зі значеннями $Z_t = i$, якщо в момент $t$ система знаходиться в стані $i$, є напівмарківським, і задається величинами $\mathrm{N, y, Q_{ij}^{(r)}(t), i, j \in S, r \in R_i}$.

З [14, 15] відомо, що напівмарківський процес називається регулярним, якщо за кінцевий проміжок часу він зі ймовірністю $p_p = 1$ перейде у будь-який стан не більше кінцевого числа разів.

Таким чином, регулярний напівмарківський процес за кінцевий проміжок часу завжди здійснює лише кінцеве число переходів.

Далі в розділі для вирішення завдання розроблення оптимізаційної стратегії напівмарківської моделі розподілу ресурсів проектування ПЗ розглядатимемо лише регулярні напівмарківські процеси.

У разі спрощенної (обмеженої одним елементом) множини рішень $R_i$ в результаті стандартних для теорії відновлення суджень отримуємо наступне рівняння відновлення з врахуванням $\alpha$ як динамічної міри оцінки витрат за часом

$$\upsilon_i(t) = (1 - H_i(t))\frac{k_i}{\alpha}(1 - e^{-\alpha t}) + \sum_{j \in S}\int_0^t \left(\frac{k_i}{\alpha}(1 - e^{-\alpha t}) + e^{-\alpha t}\upsilon_j(t - \tau)\right) dQ_{ij}(\tau), i \in S,$$

де $\upsilon_i(t)$ - спрощений запис сумарної середньої витрати $\upsilon_i(t,\beta)$ за час $t$.

В разі скінченних множин $R_i$ рівняння сумарної середньої витрати з урахуванням ймовірностей $d_i^{(r)}$ прийняття рішень про розподіл ресурсів $r$ у стані $i$ запишемо у вигляді

$$\begin{aligned} \upsilon_i(t) = & \sum_{r \in R_i} d_i^r \left(1 - H_i^{(r)}(t)\right)\frac{k_i^{(r)}}{\alpha}\left(1 - e^{-\alpha t}\right) + \\ & + \sum_{j \in S}\sum_{r \in R_i}\int_0^t d_i^r \left(\frac{k_i^{(r)}}{\alpha}\left(1 - e^{-\alpha t}\right) + e^{-\alpha t}\upsilon_j(t - \tau)\right) dQ_{ij}^{(r)}(\tau), i \in S, \end{aligned} \quad (3.9)$$

де $k_i^{(r)}$ - витрата системи за одиницю часу перебування у стані $i$ при рішенні про розподіл ресурсів $r \in R_i$;

$\upsilon_j(t)$ - сумарна середня витрата, за умови, що процес починається в момент $t=0$ зі стану $j$.

Скористаємось основними положеннями рівняння (інтеграла) Лапласа-Стілтьєса [14] для визначення величини $\upsilon_i(\beta)$, при визначеній стратегії $\beta$.

Аналіз робіт [13, 14, 20] показав, що для будь-якої функції $F(t)$, похідна $F'(t)$ якої є функцією-оригіналом, що задовольняє нерівності $F'(t)<Ce^{\alpha t}$ для всіх $t<0$, при всіх комплексних $s$, коли $\operatorname{Re} s>\alpha$ існує функція

$$F^*(s)=L_s^*\langle F(t)\rangle=\int_0^\infty e^{-st}dF(t), \tag{3.10}$$

тобто функція $e^{-st}$ при $\operatorname{Re} s>\alpha$ інтегрована за функцією $F(t)$. Функцію $F^*(s)$ називають перетворенням Лапласа-Стілтьєса функції $F(t)$.

З виразів (3.7) та (3.8) можно отримати наступне правило: $H_i^{(r)}(\infty)=1, \quad i\in S, r\in R_i$.

З цього можно стверджувати, що перша сума у виразі (3.9) при $t\to\infty$ обертається на нуль.

Інтегруючи по частинах вираз (3.10) для $L_s^*\langle F(t)\rangle$, отримуємо

$$sL_s^*\langle F(t)\rangle=L_s^*\langle F(t)\rangle-F(0), \tag{3.11}$$

де

$$F(s)=L_s\langle F(t)\rangle=\int_0^\infty e^{-st}F(t)dt$$

є перетворення Лапласа функції $F(t)$. З (3.11) при $s\neq 0$ знаходимо

$$L_s\langle F(t)\rangle=\frac{1}{s}\left(L_s^*\langle F(t)\rangle-F(0)\right). \tag{3.12}$$

Інтегруємо по частинах з урахуванням виразу (3.8), знаходимо

$$\sum_j \int_0^t \left(1-e^{-\alpha t}\right) dQ_{ij}^{(r)}(\tau) = \left(1-e^{-\alpha t}\right) \sum_j dQ_{ij}^{(r)}(\tau) \Big|_0^t - \sum_j \alpha \int_0^t e^{-\alpha t} H_i(\tau) d\tau . \qquad (3.13)$$

Проводячи перетворення, переходячи у виразі (3.13) до межі $t \to \infty$ і застосовуючи формулу (3.12) для $s = \alpha, (\alpha > 0)$, з урахуванням співвідношень (3.6) і (3.7) отримаємо

$$\sum_j \int_0^t \left(1-e^{-\alpha t}\right) dQ_{ij}^{(r)}(\tau) = (1-\alpha) L_{s=\alpha} \left\langle H_i^{(r)}(\tau) \right\rangle = 1 - \alpha \frac{1}{\alpha} L^*_{s=\alpha} \left\langle H_i^{(r)}(\tau) \right\rangle = 1 - h_i^{(r)}(\alpha),$$

де $h_i^{(r)}(\alpha) = L^*_{s=\alpha} \left\langle H_i^{(r)}(t) \right\rangle$.

Застосування для функції $\Phi_i^{(r)}(t) = \int_0^r e^{-\alpha t} \nu_j(t-\tau) dQ_{ij}^{(r)}(t)$ теореми про граничний перехід в інтегралі по параметру, від якого залежать межі інтеграції і підінтегрально функція [20], при $t \to \infty$ дозволить отримати

$$\Phi_i^{(r)}(\infty) = \int_0^\infty e^{-\alpha t} \nu_j(\alpha) dQ_{ij}^{(r)}(\tau) = \nu_j(\alpha) q_{ij}^{(r)}(\alpha), \qquad (3.15)$$

де $q_{ij}^{(r)}(\alpha) = L^*_{s=\alpha} \left\langle Q_i^{(r)}(\alpha) \right\rangle$.

Перетворюючи вираз (3.9), та визначивши межу $t \to \infty$, з урахуванням (3.14) і (3.15) отримуємо наступний аналітичний вираз:

$$\upsilon_i(t) = \sum_{r \in R_i} d_i^{(r)} \left(\zeta_i^{(r)}(\alpha)\right) + \sum_{j \in S} q_{ij}^{(r)}(\alpha) \nu_j(\alpha) \qquad (3.16)$$

де

$$\zeta_i^{(r)}(\alpha) = \frac{k_i^{(r)}}{\alpha} \left(1 - h_i^{(r)}(\alpha)\right). \qquad (3.17)$$

Нехай $\zeta_i(\alpha) = \sum_{r \in R_i} d_i^r \left(\rho_i^{(r)}(\alpha)\right)$ і $\Im(\alpha) = \left(\zeta_0(\alpha), \ldots, \zeta_N(\alpha)\right)^T$, $\wp(\alpha) = \left(\nu_0(\alpha), \ldots, \nu_N(\alpha)\right)^T$, ($T$ - символ транспонування матриці). Тоді

$$\wp(\alpha) = \Im_0(\alpha) + q(\alpha) \wp(\alpha) \qquad (3.18)$$

де $q(\alpha)=\{q_{ij}(\alpha)\}, \quad q_{ij}(\alpha)=\sum_{r\in R_i} d_i^{(r)}\left(q_{ij}^{(r)}(\alpha)\right).$

З виразу (3.18) знайдемо

$$\wp(\alpha)=\{\mathrm{I}-\mathrm{q}(\alpha)\}^{-1}\Im_0(\alpha). \tag{3.19}$$

Даний вираз справедливий, оскільки при $\alpha>0$ матриця $\{I-q(\alpha)\}$ - невироджена, $I$ - одинична матриця розміру $(N\times 1)\times(N\times 1).$

З 3.18 отримаємо наступне

$$y\upsilon(\alpha)=\sum_{i\in S}\sum_{j\in\tilde{S}}\sum_{r\in R_i} y_i\mu_{ij}(\alpha)\zeta_j^{(r)}(\alpha)d_i^{(r)}\ \{I-q(\alpha)\}^{-1}=\{\mu_{ij}(\alpha)\}. \tag{3.20}$$

З 3.20 $\mu_{ij}(\alpha)$ залежать від $d_i^{(r)}$, $r\in R_i$, $i\in S$, так як елементи матриці $\{I-q(\alpha)\}$ можна виразити через $d_i^{(r)}$, $r\in R_i$, $i\in S$.

Припустимо, що $\{d_i^{(r)}\}$ $(r\in R_i)$ - непсевдовипадкова марківська стаціонарна стратегія розподілу ресурсів розроблення ПЗ у стані $j$.

$$d_j^{(r)}\in\{0,1\},\ \sum_{j\in S} d_j^{(r)}=1,$$

і $x_{00}=1,\ x_{rj}=d_j^{(r)},\ r\in R_i,\ j\in\tilde{S}$. Мінімізація витрат розподілу ресурсів розроблення ПЗ (вираз (3.20)) дозволить сформулювати завдання оптимізації для булевих змінних $\mathrm{X}=\{x_{rj}\},\ r\in R_i,\ j\in\tilde{S}$:

$$f(\alpha,\mathrm{X})=\sum_{i\in S}\sum_{j\in\tilde{S}}\sum_{r\in R_i} y_i\mu_{ij}(\alpha,\mathrm{X})\zeta_j^{(r)}x_{rj}\to\min, \tag{3.21}$$

$$\sum_{r\in R_i} x_{rj}=1,\ j\in\tilde{S}, \tag{3.22}$$

$$\sum_{j\in\tilde{S}} c_{rj}x_{rj}\le b_r,\ r\in R_i,\ j\in\tilde{S}, \tag{3.23}$$

$$x_{rj}\in\{0,1\},\ j\in\tilde{S},\ r\in R_i. \tag{3.24}$$

## 3.3 Розроблення оптимізаційної марківської стаціонарної стратегії розподілу ресурсів проектування безпечного ПЗ з лінійними обмеженнями

Наведене вище завдання оптимізації для булевих змінних $\mathrm{X} = \{x_{rj}\}$, (3.21), (3.24) включає до себе множину псевдобулевих нерівностей (3.23), (3.24).

Додаткова умова (3.22) розширює межі системи, при цьому можна записати, що $\mathrm{X}_r^{(k)} = \{x_{r1}^{(k)}, \ldots, x_{rN}^{(k)}\}$, $k = 1, \ldots k_r$ - множина допустимих рішень $r$-ої нерівності запропонованої системи.

Для побудови рішень вдосконаленої системи при відомих допустимих рішеннях $\mathrm{X}_r^{(k)} = \{x_{r1}^{(k)}, \ldots, x_{rN}^{(k)}\}$, $k = 1, \ldots k_r$ можливо використання наступних обмежень.

Представимо рішення вдосконаленої системи як $Z = \{s_j\}$, $j = 1, \ldots N$, де $s_j$ - множина номерів $r$, для яких допустимо обмеження $x_{rj} = 1$.

Для рішення цієї системи потрібно $m$ кроків, де $m$ - число обмежень (3.23). У початковому стані кожна з $s_j^{(0)}$ вектору $\mathrm{Z}^{(o)}$ включає усі можливі значення $r \in R_i$.

На $r$-му кроці відбувається перетин вектору $\mathrm{Z}^{(r-1)}$ з одним з рішень множини $r$-ої нерівності.

Допускаючи, що умови $r$-ій нерівності відповідає $r = r_1$, а також, що $\alpha_j$ є $j$-м елементом множини допустимих рішень $r$-ої нерівності, $\alpha_j \in \{0,1,\varphi\}$, де $\varphi$ - невизначений параметр з множини $\{0,1\}$, сформулюємо обмеження для $r$-го кроку алгоритму побудови рішень вдосконаленої системи.

1. Якщо значення $\alpha_j$ не фіксоване, то $s_j^{(r)} = s_j^{(r-1)}$.

2. Якщо $\alpha_j = 1$, то при $r_1 \in s^{(r-1)}$ допускаємо $s^{(r)} = \{r_1\}$, а при $r_1 \overline{\in} s^{(r-1)}$ допускаємо $s^{(r)}$ дорівнює пустій множині.

3. Якщо $\alpha_j = 0$, то допускаємо $s^{(r)} = s^{(r-1)} / \{r_1\}$.

Пошук рішень розподілу ресурсів розроблення ПЗ здійснюється з урахуванням (3.22).

На *m* кроці алгоритму реалізується вектор $Z^{(m)} = \left\{\alpha_1^{(m)}, \ldots, \alpha_N^{(m)}\right\}$, кожна компонента $\alpha_j^{(m)}$, якого є найпростішою множиною $\{r\}$, $r \in R$, $R = \{1, \ldots, m\}$, і отже, $Z^{(m)}$ є рішенням вдосконаленої системи.

Якщо з допомогою вдосконаленої системи існує можливість отримання декількох рішень розподілу ресурсів розроблення ПЗ необхідно знайти сукупність усіх рішень вдосконаленої системи, та обрати з них оптимальне рішення, що доставляє мінімум цільової функції $f(\alpha, \mathrm{X})$.

Це рішення може знаходитися різними відомими методами лінійного програмування, або просто шляхом безпосереднього порівняння значень $f(\alpha, \mathrm{X})$ при визначенні X вдосконаленої системи.

Розглянемо практичний приклад реалізації методу оптимізації розподілу ресурсів розроблення програмного забезпечення в умовах підвищених вимог щодо захисту інформації для напівмарківської моделі прийняття рішень.

## 3.4 Практичні рекомендації застосування методу оптимізації розподілу ресурсів розроблення безпечного програмного забезпечення в умовах підвищених вимог щодо захисту інформації

В процесі експлуатації системи, в разі знаходження системи у стані взарливості (рис. 3.1), відбуваються втрати, які пропорційні часу перебування системи в цьому стані.

Також до цих витрат приєднуються витрати на заходи, які призначено для повернення системи у нормальний стан (рис. 3.2). Прийняті рішення по усуванню вразливостей теж вимагають витрат ресурсів, які є пропорційні часу їх застосування.

Час проведення відновлювальних заходів задано відповідними розподілами зміни стану в часі. Позначимо як β стратегію прийняття тих чи інших рішень у вигляді множини псевдобулевих значень – застосовується випадкове рішення або ні.

Ігнорування ситуації є теж рішенням, при якому система може повернутися у нормальний стан але за значно більший час та значних втратах.

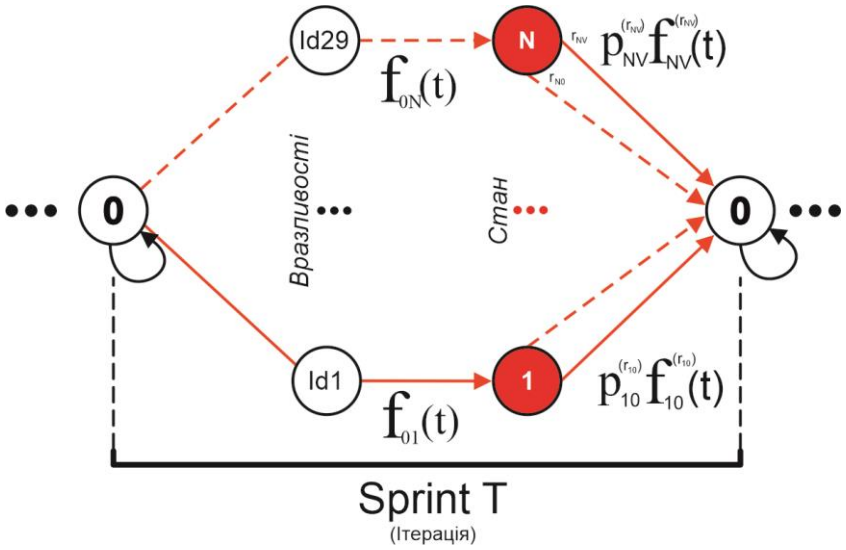


Рис. 3.1. Загальна схема оптимізації розподілу ресурсів

Деякі ситуації не можуть мати пункту ігнорування, бо такий підхід приводить розробку проекту до заморозки.

$Q_{ij}(t)$ – ймовірність стану *j* в момент *t*, якщо початковий стан є *i*. Тоді за час [0;t] середня ймовірність $P_{ij}$ є $P_{ij} = \frac{1}{t}\int_0^t Q_{ij}(\tau)\cdot d\tau$. В стаціонарному режимі системи $Q_{ij}(t)$ є мало змінною, тому $P_{ij} \approx Q_{ij}(t)$ коли $t \to \infty$.

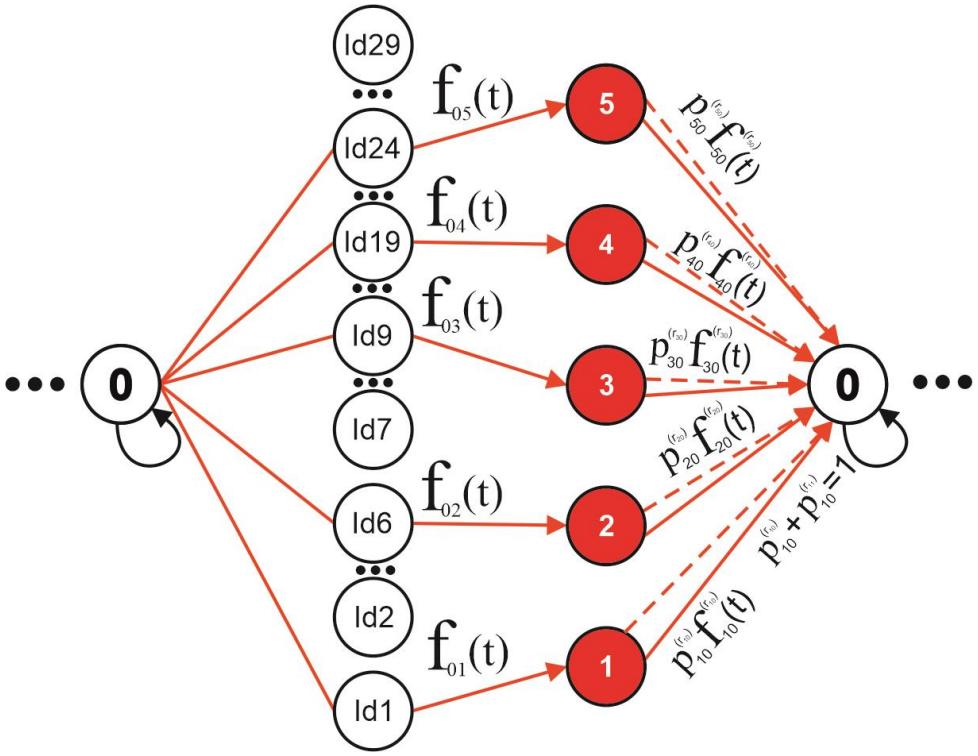


Рис. 3.2. Схема прикладу оптимізації розподілу ресурсів розроблення безпечного ПЗ

Відповідно (3.2) необхідно мінімізувати витрати $M \to \min$ для цього необхідно визначити наступні функції.

$$\begin{cases} Q_{ij}(t) = \sum\limits_{k\in\{s/i\}} \sum\limits_{r\in R_{ij}} \int\limits_0^t p_{ik}^{(r)} f_{ik}^{(r)}(\tau)\cdot Q_{kj}(t-\tau)\cdot d\tau; i \neq j; i, j \in S \\ Q_{ii}(t) = (1 - \sum\limits_{k\in\{s/i\}} \sum\limits_{r\in R_{ik}} \int\limits_0^t p_{ik}^{(r)} f_{ik}^{(r)}(\tau) d\tau) + \sum\limits_{k\in\{s/i\}} \sum\limits_{r\in R_{ik}} \int\limits_0^t p_{ik}^{(r)} f_{ik}^{(r)}(\tau)\cdot Q_{kj}(t-\tau)\cdot d\tau); i = j; i \in S \end{cases} \quad (3.25)$$

Позначимо перетворення Лапласа для більш коротких подальших записів:

$$f_{ij}^{*(r)} \Leftrightarrow L\left\langle f_{ij}^{(r)}(t)\right\rangle \Leftrightarrow f_{ij}^{*(r)}(s)$$

$$Q_{ij}^{*(r)} \Leftrightarrow L\left\langle Q_{ij}^{(r)}(t)\right\rangle \Leftrightarrow Q_{ij}^{*(r)}(s)$$

За властивостями перетворення система (3.25) перетворюється у наступну:

$$\begin{cases} Q_{ij}^{*} = \sum\limits_{k\in\{s/i\}}\sum\limits_{r\in R_{ik}}\int\limits_0^t p_{ik}^{(r)} f_{ik}^{*(r)} Q_{kj}^{*}; i\neq j; i,j\in M \\ Q_{ii}^{*} = (1-\sum\limits_{k\in\{s/i\}}\sum\limits_{r\in R_{ik}} p_{ik}^{(r)} f_{ik}^{*(r)})\frac{1}{s} + \sum\limits_{k\in\{s/i\}}\sum\limits_{r\in R_{ik}} p_{ik}^{(r)} f_{ik}^{*(r)} Q_{ki}^{*}; i=j; i,j\in M \end{cases} \quad (3.26)$$

З системи маємо розв'язки: $\left\{Q_{0i}^{*} = Q_{0i}^{*(r)}(S); i\in S\right.$

Для конкретного набору R будуємо таблицю (табл. 3.1) та зафіксуємо параметри процесу та розглянемо інтенсивність потоків вразливостей (табл. 3.2).

Таблиця 3.1. Вибір значень відповідно стратегії

| R | № рішення | $p_{i0}^{(r)}$ значення ймовірностей | | Обрані значення |
|---|---|---|---|---|
| | | I | II | |
| $1\to 0$ | $r_{10}$ | 0 | 1 | $p_{10}^{(r_{11})}=0$ |
| | $r_{11}$ | 1 | 0 | $p_{10}^{(r_{11})}=1$ |
| $2\to 0$ | $r_{20}$ | 0 | 1 | $p_{20}^{(r_{20})}=1$ |
| | $r_{21}$ | 1 | 0 | $p_{20}^{(r_{21})}=0$ |
| $3\to 0$ | $r_{30}$ | 0 | 1 | $p_{30}^{(r_{30})}=0$ |
| | $r_{31}$ | 1 | 0 | $p_{30}^{(r_{31})}=1$ |
| $4\to 0$ | $r_{40}$ | 0 | 1 | $p_{40}^{(r_{40})}=1$ |
| | $r_{41}$ | 1 | 0 | $p_{40}^{(r_{41})}=0$ |
| $5\to 0$ | $r_{50}$ | 0 | 1 | $p_{50}^{(r_{50})}=1$ |
| | $r_{51}$ | 1 | 0 | $p_{50}^{(r_{51})}=0$ |

Таблиця 3.2. Інтенсивність потоків вразливостей

| № | Назва вразливості | Стан | Поток вразливостей $\lambda_{0i}$ |
|---|---|---|---|
| 0 | Стан відсутності вразливості (без прецедентів) | 0 | - |
| Id1 | Нерозуміння або невиконання стратегії розроблення ПЗ | 1 | 0,3 |
| Id6 | Неефективний менеджмент проектів | 2 | 0,15 |
| Id24 | Невиявлення загроз безпеки ПЗ | 3 | 0,1 |
| Id19 | Неадекватний менеджмент існуючих систем | 4 | 0,4 |
| Id9 | Недостатній супровід з боку менеджерів | 5 | 0,15 |

Таблиця 3.3. Характеристики прийнятих рішень

| № | Стан | Можливі варіанти рішення вразливостей | Позначення рішення | Збитки за спринт (*у.о.*) | Середній час заходу у спринтах |
|---|---|---|---|---|---|
| Id1 | 1 | Виділення часу ПМ для уточнення подальшої стратегії розроблення ПЗ | $r_{10}$ | 80 | 1 |
| | | Робота всієї команди в повному составі для уточнення стратегі розроблення ПЗ | $r_{11}$ | 600 | 0,5 |
| Id6 | 2 | Виділення додаткового часу ПМ. | $r_{20}$ | 60 | 0,25 |
| | | Сторонній найм перекладача ТЗ. | $r_{21}$ | 300 | 0,1 |
| Id24 | 3 | Документування можливої вразливості ПМ. | $r_{30}$ | 25 | 0,05 |
| | | Сторонній найм аутсорс тестувальника ПЗ – Midle Penetration Tester | $r_{31}$ | 300 | 1 |
| Id19 | 4 | Виділення часу ПМ для вирішення проблеми API взаємодії зі стороннім сервісом. | $r_{40}$ | 200 | 0,35 |
| | | Виділення коштів розробнику на платну консультацію з представником сервісу. | $r_{41}$ | 100 | 0,25 |
| Id9 | 5 | Зняти навантаження зі стороннього проекту з ПМ. | $r_{50}$ | 100 | 1 |
| | | Виділення додаткових овертайм (overtime) коштів ПМ. | $r_{51}$ | 200 | 0,5 |

Таблиця 3.4. Інтенсивність потоків при переході з урахуванням прийнятого рішення $f_{ij}^{(r)}(t) = \lambda_{ij}^{(r)} e^{-\lambda_{ij}^{(r)} t}$

| i – було \ j – стало | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | | $\frac{1}{5} 0{,}3 e^{-0{,}3t}$ | $\frac{0{,}15}{5} e^{-0{,}15t}$ | $\frac{0{,}1}{5} e^{-0{,}1t}$ | $\frac{0{,}4}{5} e^{-0{,}4t}$ | $\frac{0{,}15}{5} e^{-0{,}15t}$ |
| 1 | $r_{10} \Rightarrow 1e^{-1t}$<br>$r_{11} \Rightarrow 2e^{-2t}$ | | 0 | 0 | 0 | 0 |
| 2 | $r_{20} \Rightarrow 4e^{-4t}$<br>$r_{21} \Rightarrow 10e^{-10t}$ | 0 | | 0 | 0 | 0 |
| 3 | $r_{30} \Rightarrow 20e^{-20t}$<br>$r_{31} \Rightarrow 1e^{-t}$ | 0 | 0 | | 0 | 0 |
| 4 | $r_{40} \Rightarrow \frac{20}{7} e^{-\frac{20}{7}t}$<br>$r_{41} \Rightarrow 4e^{--4t}$ | 0 | 0 | 0 | | 0 |
| 5 | $r_{50} \Rightarrow 1e^{-t}$<br>$r_{51} \Rightarrow 2e^{-2t}$ | 0 | 0 | 0 | 0 | |

Середній час заходу дозволить визначити відповідні густини розподілів часу переходів між станами системи.

Результат показано в таблиці 3.4. Потоки повернення в стан 0 в залежності від обраного рішення $r_{ij}$.

Після визначення параметрів системи, маємо можливість отримати з системи 3.27 наступну систему:

$$
\begin{cases}
Q_{00}^{*} = \dfrac{0{,}08Q_{40}^{*}}{s+0{,}4} + \dfrac{1-\dfrac{0{,}08}{s+0{,}4}-\dfrac{0{,}06}{s+0{,}3}-\dfrac{0{,}06}{s+0{,}15}-\dfrac{0{,}02}{s+0{,}1}}{s} + \\
+ \dfrac{0{,}06Q_{10}^{*}}{s+0{,}3} + \dfrac{0{,}03Q_{50}^{*}}{s+0{,}15} + \dfrac{0{,}03Q_{20}^{*}}{s+0{,}15} + \dfrac{0{,}02Q_{30}^{*}}{s+0{,}1}; \\
Q_{10}^{*} = \dfrac{2Q_{00}^{*}p_{10}^{(r_{11})}}{s+2} + \dfrac{Q_{00}^{*}p_{10}^{(r_{10})}}{s+1}; \\
Q_{20}^{*} = \dfrac{10Q_{00}^{*}p_{20}^{(r_{21})}}{s+10} + \dfrac{4Q_{00}^{*}p_{20}^{(r_{20})}}{s+4}; \\
Q_{30}^{*} = \dfrac{20Q_{00}^{*}p_{30}^{(r_{30})}}{s+20} + \dfrac{Q_{00}^{*}p_{30}^{(r_{31})}}{s+1}; \\
Q_{40}^{*} = \dfrac{4Q_{00}^{*}p_{40}^{(r_{41})}}{s+4} + \dfrac{20Q_{00}^{*}p_{40}^{(r_{40})}}{7s+20}; \\
Q_{50}^{*} = \dfrac{2Q_{00}^{*}p_{50}^{(r_{51})}}{s+2} + \dfrac{Q_{00}^{*}p_{50}^{(r_{50})}}{s+1};
\end{cases}
$$

Аналітичні розв’язки отримані за допомогою системи аналітичних перетворень математичного пакету для *wxMaxima* та наведені у додатку Б.

У результаті маємо:

$$
\begin{cases}
Q_{00}^{*}(s, p_{10}^{(r10)}, p_{10}^{(r11)}, p_{20}^{(r20)}, p_{20}^{(r21)}, p_{30}^{(r30)}, p_{30}^{(r31)}, p_{40}^{(r40)}, p_{40}^{(r41)}, p_{50}^{(r50)}, p_{50}^{(r51)}) \\
Q_{10}^{*}(s, p_{10}^{(r10)}, p_{10}^{(r11)}, p_{20}^{(r20)}, p_{20}^{(r21)}, p_{30}^{(r30)}, p_{30}^{(r31)}, p_{40}^{(r40)}, p_{40}^{(r41)}, p_{50}^{(r50)}, p_{50}^{(r51)}) \\
Q_{20}^{*}(s, p_{10}^{(r10)}, p_{10}^{(r11)}, p_{20}^{(r20)}, p_{20}^{(r21)}, p_{30}^{(r30)}, p_{30}^{(r31)}, p_{40}^{(r40)}, p_{40}^{(r41)}, p_{50}^{(r50)}, p_{50}^{(r51)}) \\
Q_{30}^{*}(s, p_{10}^{(r10)}, p_{10}^{(r11)}, p_{20}^{(r20)}, p_{20}^{(r21)}, p_{30}^{(r30)}, p_{30}^{(r31)}, p_{40}^{(r40)}, p_{40}^{(r41)}, p_{50}^{(r50)}, p_{50}^{(r51)}) \\
Q_{40}^{*}(s, p_{10}^{(r10)}, p_{10}^{(r11)}, p_{20}^{(r20)}, p_{20}^{(r21)}, p_{30}^{(r30)}, p_{30}^{(r31)}, p_{40}^{(r40)}, p_{40}^{(r41)}, p_{50}^{(r50)}, p_{50}^{(r51)}) \\
Q_{50}^{*}(s, p_{10}^{(r10)}, p_{10}^{(r11)}, p_{20}^{(r20)}, p_{20}^{(r21)}, p_{30}^{(r30)}, p_{30}^{(r31)}, p_{40}^{(r40)}, p_{40}^{(r41)}, p_{50}^{(r50)}, p_{50}^{(r51)})
\end{cases}
$$

Для подальшого аналізування є необхідність визначитися з ймовірностями прийняття рішення $p_{i0}^{(ri0)}, p_{i0}^{(ri1)}$ (табл.3.1). Ці ймовірності задають стратегію прийняття рішень.

З метою демонстрації оцінювання конкретної стратегії, зупинимося на двох стратегіях:

$$
\beta_0: \ [p_{10}^{(r10)} = 0; p_{10}^{(r11)} = 1; p_{20}^{(r20)} = 0; p_{20}^{(r21)} = 1; p_{30}^{(r30)} = 0; p_{30}^{(r31)} = 1;
$$

$$
p_{40}^{(r40)} = 1; p_{40}^{(r41)} = 0; p_{50}^{(r50)} = 0; p_{50}^{(r51)} = 1];
$$

$$
\beta_1: \ [p_{10}^{(r10)} = 0; p_{10}^{(r11)} = 1; p_{20}^{(r20)} = 1; p_{20}^{(r21)} = 0; p_{30}^{(r30)} = 0; p_{30}^{(r31)} = 1;
$$

$$p_{40}^{(r40)} = 1; p_{40}^{(r41)} = 0; p_{50}^{(r50)} = 0; p_{50}^{(r51)} = 1].$$

У додатку Б ці пробні стратегії позначені як *test00* та *test01* відповідно.

Тоді:

$$Q_{00}^{*}(s) = \frac{875}{953 \cdot s} + \frac{27300000 \cdot s^6 + 221654000 \cdot s^5 + 587615600 \cdot s^4 + 589473100 \cdot s^3 + }{953 \cdot (350000 \cdot s^7 + 3782500 \cdot s^6 + 15094000 \cdot s^5 + 27698025 \cdot s^4 + 23967780 \cdot s^3 + }$$
$$\frac{+ 211500325 \cdot s^2 + 29590650 \cdot s + 2119200}{+ 9427076 \cdot s^2 + 1601232 \cdot s + 91488)}$$

$$Q_{01}^{*}(s) = \frac{15}{953 \cdot s} - \frac{5250000 \cdot s^6 + 36724500 \cdot s^5 + 56156550 \cdot s^4 + 54992370 \cdot s^3 + }{953 \cdot (350000 \cdot s^7 + 3782500 \cdot s^6 + 15094000 \cdot s^5 + 27698025 \cdot s^4 + 23967780 \cdot s^3 + }$$
$$\frac{+ 125275353 \cdot s^2 + 50289810 \cdot s + 4480032}{+ 9427076 \cdot s^2 + 1601232 \cdot s + 91488)}$$

$$Q_{02}^{*}(s) = \frac{15}{1906 \cdot s} - \frac{5250000 \cdot s^6 + 36724500 \cdot s^5 + 93180600 \cdot s^4 + 106326705 \cdot s^3 + }{1906 \cdot (350000 \cdot s^7 + 3782500 \cdot s^6 + 15094000 \cdot s^5 + 27698025 \cdot s^4 + 23967780 \cdot s^3 + }$$
$$\frac{+ 53392134 \cdot s^2 + 8313972 \cdot s + 248712}{+ 9427076 \cdot s^2 + 1601232 \cdot s + 91488)}$$

$$Q_{03}^{*}(s) = \frac{30}{953 \cdot s} - \frac{10500000 \cdot s^6 + 106804000 \cdot s^5 + 388063650 \cdot s^4 + 611488675 \cdot s^3 + }{953 \cdot (350000 \cdot s^7 + 3782500 \cdot s^6 + 15094000 \cdot s^5 + 27698025 \cdot s^4 + 23967780 \cdot s^3 + }$$
$$\frac{+ 410570172 \cdot s^2 + 114401932 \cdot s + 10709856}{+ 9427076 \cdot s^2 + 1601232 \cdot s + 91488)}$$

$$Q_{04}^{*}(s) = \frac{21}{1906 \cdot s} - \frac{7350000 \cdot s^6 + 26064500 \cdot s^5 + 85954400 \cdot s^4 + 375763395 \cdot s^3 + }{953 \cdot (350000 \cdot s^7 + 3782500 \cdot s^6 + 15094000 \cdot s^5 + 27698025 \cdot s^4 + 23967780 \cdot s^3 + }$$
$$\frac{+ 368416216 \cdot s^2 + 105775376 \cdot s + 8161272}{+ 9427076 \cdot s^2 + 1601232 \cdot s + 91488)}$$

$$Q_{05}^{*}(s) = \frac{15}{953 \cdot s} - \frac{5250000 \cdot s^6 + 46731000 \cdot s^5 + 139782300 \cdot s^4 + 167695140 \cdot s^3 + }{953 \cdot (350000 \cdot s^7 + 3782500 \cdot s^6 + 15094000 \cdot s^5 + 27698025 \cdot s^4 + 23967780 \cdot s^3 + }$$
$$\frac{+ 83717547 \cdot s^2 + 14209230 \cdot s + 94368}{+ 9427076 \cdot s^2 + 1601232 \cdot s + 91488)}$$

Перші доданки є образами Лапласа сталих в часі. Знаменники других доданків містять один й той самий поліном:

$$350000 \cdot s^7 + 3782500 \cdot s^6 + 15094000 \cdot s^5 + 27698025 \cdot s^4 + 23967780 \cdot s^3 +$$
$$9427076 \cdot s^2 + 1601232 \cdot s + 91488$$

Корені цього поліному мають від'ємну дійсну частину:

$[s_1 \approx -0{,}116; s_2 \approx -0{,}247; s_3 \approx -0{,}364; s_4 \approx -1{,}015; s_5 \approx -2{,}082;$

$s_6 \approx -2{,}950; \quad s_7 \approx -4{,}034;]$

Тому дріб розкладається в суму простих дробів $A_i / s - s_i$, які є образами експонентного затухання:

$$\lim_{t\to\infty} A_i e^{s_i t} = 0$$

Тоді образи констант є значенням ймовірності для стаціонарної системи:

$$P_0 = \frac{875}{953};\ P_1 = \frac{15}{953};\ P_2 = \frac{15}{1906};\ P_3 = \frac{30}{953};\ P_4 = \frac{21}{1906};\ P_5 = \frac{15}{953};$$

Легко пересвідчитися, що сума цих ймовірностей рівна 1. Відповідно до суми витрат, за один спринт (80 годин) середні витрати складатимуть:

$$\begin{cases} M = P_1(Z_1 + B_1^{(r_{11})}) + P_2(Z_2 + B_2^{(r_{21})}) + P_3(Z_3 + B_3^{(r_{31})}) + P_4(Z_4 + B_4^{(r_{40})}) + P_5(Z_5 + B_5^{(r_{51})}) \\ M = \frac{15}{935}\cdot 600 + \frac{15}{1906}\cdot 300 + \frac{30}{953}\cdot 300 + \frac{21}{1906}\cdot 200 + \frac{15}{953}\cdot 200 = 26{,}78 \end{cases}$$

Для стратегії $\beta_1$ зразу випишемо ймовірності для стаціонарної системи:

$$P_0 = \frac{250}{271};\ P_1 = \frac{30}{1897};\ P_2 = \frac{6}{1897};\ P_3 = \frac{60}{1897};\ P_4 = \frac{3}{271};\ P_5 = \frac{30}{1897};$$

Тоді середні витрати за спринт (80 годин) складатимуть:

$$M = \frac{30}{1877}\cdot 600 + \frac{6}{18973}\cdot 60 + \frac{60}{1897}\cdot 300 + \frac{3}{271}\cdot 200 + \frac{30}{1897}\cdot 200 \approx 25{,}3$$

Середні витрати 25.3<26.6, тому стратегія $\beta_1$ є більш прийнятною. Слід зауважити, що розроблена метод має можливість враховувати ситуації, коли більш вигідну стратегію завжди використовувати не є можливим. Тоді нехай більш вигідне рішення $r_{21}$ використовується з ймовірністю $p_2^{(r_{21})} = 0.9$, а менш вигідне $p_2^{(r_{20})} = 0.1$ (в додатку Б, *test02*):

$$P_0 = \frac{17500}{18979};\ P_1 = \frac{300}{18979};\ P_2 = \frac{69}{18979};\ P_3 = \frac{600}{18979};\ P_4 = \frac{210}{18979};\ P_5 = \frac{300}{18979};$$

Для $P_2$ витрати складатимуть 60 у 0,1 долі випадків та 300 в 0,9 долі випадків. Тому загальні витрати на sprint складатимуть:

$$M = \frac{300}{18979}\cdot 600 + \frac{69}{18979}\cdot (60\cdot 0{,}1 + 300\cdot 0{,}9) + \frac{600}{18979}\cdot 300 + \frac{210}{18979}\cdot 200 + \frac{300}{18979}\cdot 200 \approx 25{,}35\,.$$

Відповідно до завдання $M \to \min$, послідовно обираються інші доступні рішення, які дають можливість зменшення середніх витрат. Іноді вибір рішень потрібно повторити ітеративно кілька раз, бо функція втрат не є лінійною.

Звісно, стратегія з ймовірністю прийняти не оптимальне рішення не є більш прибутковою, але тут враховано ймовірність відсутності засобів використати оптимальне рішення, що в свою чергу дає змогу обрати не лише крашу стратегію, але й оцінити реальні спроможності при прийняті рішень.

## 3.5 Висновки до розділу

У розділі удосконалено метод оптимізації розподілу ресурсів розроблення програмного забезпечення в умовах підвищених вимог щодо захисту іннформації. В основу цього методу була покладена напівмарківська модель розподілу ресурсів проектування ПЗ для керованого марківського процесу у безперервному часі.

Відмінною особливістю запропонованого методу є використання псевдобулевих методів бівалентного програмування з цільовою функцією, що має нелінійні властивості, і лінійними обмеженнями для визначення оптимальної стратегії усунення експлуатаційних помилок.

Проведені дослідження показали, що використовувані в даному розділі теоретичні положення в достатньому об'ємі відбивають стандарти і можливості сучасних методологій тестування ПЗ.

У якості прикладу розглянуто ситуації виникнення помилок безпеки ПЗ, і визначено оптимальну стратегію оптимізації для усунення вказаної аномальної ситуації.

Слід зауважити, що представлений в розділі метод доцільно використати не лише при управлінні вразливостями розробки ПЗ, але і при

функціональному, навантажувальному, стресовому та інших видах тестування для запобігання можливим втратам.

Отпубліковані наукові результати роботи автора приведені у 3 розділі [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34].

**Список використаних джерел:**


[1] Stoneburner Gary Goguen, Alice, and Feringa Alexis Risk Management Guide for Information Technology Systems Recommendations of the National Institute of Standards and Technology // Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, 2002. - 55 p.

[2] Zeng Y. Risk Management For Enterprise Resource Planning System Implementations in Project-Based / Y. Zeng // Firms : dis. for the degree of PHD, Maryland, 2010. – P. 210.

[3] Seacord Robert Secure Coding in C and C++ / Robert C. Seacord – Addison-Wesley, 2013. – 600 p.

[4] Seacord Robert The CERT® C Coding Standard, Second Edition: 98 Rules for Developing Safe, Reliable, and Secure / Robert C. Seacord – Addison-Wesley, 2014. - pp 576.

[5] Seacord Robert The CERT C Secure Coding Standard / Robert C. Seacord – Addison-Wesley, 2008. - pp 720.

[6] Макконнелл С. Совершенный код. Мастер-класс. М.: Рус. Редакция ; СПб.: Питер, 2007. 896 с.

[7] Вендров А.М. Проектирование программного обеспечения экономических информационных систем / А.М. Вендров. – М.: Финансы и статистика, 2005. – 544 с.

[8] Krishnan M. Soumya Software Development Risk Aspects and Success Frequency on Spiral and Agile Model / M. Soumya Krishnan // International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization). – 2015. –Vol.3. – №1. – pp.301-310

[9] Sherman M. Building Secure Software for Mission Critical Systems [Електронний ресурс] – Режим доступу: http://resources.sei.cmu.edu/asset_files/Presentation/2017_017_001_495865.pdf

[10] Благодатских В.А. Стандартизация разработки программных средств / В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов – М: Финансы и статистика, 2003. – 288 с.

[11] Гусятников В.Н. Стандартизация и разработка программных систем / В.Н. Гусятников, А.И. Безруков – М: Финансы и статистика, 2010. – 288 с.

[12] Лужецький В.А. Організаційно-правові питання безпеки інформації Концептуальна модель системи інформаційного впливу / В.А. Лужецький // Безпека інформації. Ukrainian Scientific Journal of Information Security Том 23, № 1 (2017).

[13] Саати Т.Л. Принятие решений при зависимостях и обратных связях: Аналитические сети. Пер. с англ./ Т.Л. Саати Изд. 5, стереотип. URSS – 2019. – 360 с.

[14] Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

[15] Вентцель Е.С. Теория вероятностей: Учеб. для вузов. – 6-е изд. стер. – М.: Высш. шк., 1999.– 576 с.

[16] Markou, M. Novelty detection: a review – part 1: statistical approaches /

M. Markou, S. Singh // Signal processing. – 2003. – Vol. 83, №. 12. – pp. 2481-2497.

[17] Markou, M. Novelty detection: a review – part 2: neural network based approaches / M. Markou, S. Singh //Signal processing. – 2003. – Vol. 83, №. 12. – pp. 2499-2521.

[18] Rabiner L.R., A tutorial on Hidden Markov Models and selected applications in speech recognition, in Proceedings of the IEEE 77 (2): 257–286, 1989.

[19] Smyth, P. Clustering sequences with hidden Markov models / P. Smyth : Advances in neural information processing systems. – 1997. – P. 648-654.

[20] Дышин О.А. Полумарковские модели управления рисками в магистральных газонефтетрубопроводных системах // О.А. Дышин, И.А. Азизов [Електронний ресурс]. – Режим доступу до ресурсу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/12809/02-Dishin.pdf ?sequence=1

[21] Постанова Кабінета Міністрів України від 29.03.2006 №373 «Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах» [Електронний ресурс]. – Режим доступу до ресурсу: http://zakon2.rada.gov.ua/laws/show/373-2006-п

[22] Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

[23] Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко // Системи управління, навігації та

зв’язку. – Випуск 2 (38). – Полтава: ПолтНТУ. – 2016. – С. 93-100.

[24] Коваленко О.В. Управління ризиками розробки програмного забезпечення за умови обмеженості коштів виділених на усунення помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький: ЦНТУ. - 2018. – С. 128-140.

[25] Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

[26] Коваленко О.В. Удосконалений метод управління ризиками розробки програмного забезпечення на основі напівмарковської моделі прийняття рішень/ О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2 (3 ). – Харків. – 2018. – С. 48-41.

[27] Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

[28] Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Кібербезпека в Україні: правові та організаційні питання». м. Одеса, 21 жовтня 2016 р. – Одеса : ОДУВС, 2016. – С.146-148.

[29] Коваленко А.В. Метод управления рисками разработки программного

обеспечения с использованием псевдобулевых методов бивалентного программирования / А.А. Смирнов, А.В. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Актуальні задачі та досягнення у галузі кібербезпеки». м. Кропивницький, 23-25 листопада 2016 року – Кропивницький: ЦНТУ, 2016. – С. 162.

[30] Коваленко А.В. Псевдобулевые методы бивалентного программирования для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко, С.А. Смирнов // Збірник наукових праць III міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 22-25 лютого 2017 р. – Київ: Європейський університет. – 2017. – С. 158-162.

[31] Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез II науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 23-24 березня 2017 р. – Київ: КНУ ім. Тараса Шевченко – 2017. – С. 203-205.

[32] Коваленко А.В. Метод управления рисками разработки программного обеспечения на основе алгоритмов анализа уязвимостей / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 20-22 квітня 2017 р. – Кіровоград: КНТУ. – 2017. – С. 92.

[33] Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». *CEUR Workshop Proceedings* Volume 2588, 2019,

Pages 567-579. Режим доступу: https://www.scopus.com/ record/ display.uri?eid=2-s2.0-85083214331& origin= AuthorNamesList& txGid= 176e2cada8976a6e0f3d5633fbcd3fbe **(Scopus).**

[34] Коваленко А.В. Использование псевдобулевых методов бивалентного программирования для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

# РОЗДІЛ 4
# КОМПЛЕКС МАТЕМАТИЧНИХ МОДЕЛЕЙ ПРОЦЕСУ ТЕСТУВАННЯ WEB-ЗАСТОСУНКІВ

У розділі представлено результати дослідження і алгоритми тестування на вразливість до одних з найбільш поширених видів атак на *Web*-застосунки - *DOM XSS* і *SQL*-ін'єкції.

Розроблено комплекс математичних моделей процесу тестування *Web*-застосунків. В основу математичного моделювання покладено підхід мережевого *GERT* моделювання. В результаті розроблено математичні моделі технології тестування комплексу *DOM XSS* вразливостей і технології тестування вразливості до *SQL*-ін'єкцій.

Математична модель технології тестування комплексу *DOM XSS* вразливостей відрізняється від відомих урахуванням специфіки комплексного аналізу різних типів *XSS* уразливості ("*stored XSS*", "*reflected XSS*" і *DOM Based XSS*), а також включенням в алгоритм процедур автоматичного аудиту *DOM Based XSS* окремо.

Математична модель технології тестування вразливості до *SQL*-ін'єкцій відрізняється від відомих вдосконаленим способом визначення відстані між результатами ін'єкції.

## 4.1 Математична модель технології тестування комплексу *DOM XSS* вразливостей

### 4.1.1 Алгоритм виявлення комплексу *DOM XSS* вразливостей

Збільшення кількості користувачів всесвітньої мережі Інтернет, постійне зростання інформаційного, фінансового і ділового контенту в

кіберпросторі обумовлює підвищення попиту на *Web*-застосунки. В той же час, цей процес викликає зворотну негативну реакцію з боку зловмисників, що мають постійну можливість аналізу об'єктивно існуючих вразливостей Інтернет-застосунків.

Аналіз різного роду статистичних матеріалів відомих організацій (наприклад, *Open Web Application Security Project*) показав, що одним з найбільш небезпечних видів атак (вразливостей) є міжсайтовий скриптінг - *XSS* (*Cross Site Scripting*).

Аналіз літератури [1-8] показав, що міжсайтовий скриптінг - це помилка валідації користувацьких даних, яка дозволяє передати *JavaScript* код на виконання у браузер користувача. В широких колах фахівців такі атаки часто також називають *HTML*-ін'єкціями, оскільки механізм їх впровадження дуже схожий з *SQL*-ін'єкціями, але, на відміну від останніх, впроваджуваний код виконується у браузері користувача.

З робіт [9-12] відомо, що під *XSS* зазвичай мається на увазі моментальний [13] і відкладений [14] міжсайтовий скриптінг. При моментальному *XSS* зловмисний код (*JavaScript*) повертається атакованим сервером негайно в якості відповіді на *HTTP*-запит. Відкладений *XSS* означає, що зловмисний код зберігається в атакованій системі, і пізніше може бути впроваджений в *HTML*-сторінку вразливої системи. Така класифікація припускає, що фундаментальна властивість *XSS* полягає в тому, що зловмисний код відсилається з браузеру на сервер і повертається в цей же браузер (моментальний *XSS*) або будь-який інший браузер (відкладений *XSS*).

У ряді інтернет-статей [12, 15-19] детально описано основні механізми виникнення подібного роду загроз, а також шляхи можливого блокування. Проте, щоб ідентифікувати ці загрози і можливі наслідки їх поширення в процесі безпечного управління IT-проектами, а також запропонувати

оптимальні шляхи вирішення цієї проблеми, існує необхідність математичної формалізації процесу їх ініціалізації і поширення.

У ряді робіт здійснено спроби математичної формалізації процесу пошуку і усунення вразливостей подібного роду. Так, в роботах [12, 16, 18] представлено узагальнені матеріали механізмів і процедур безпечного програмування, переслідуючих цілі зниження вразливостей. В роботах [12, 15, 17, 19] представлено математичні моделі, що описують алгоритми аналізу *Web*-застосунків (у тому числі і алгоритм однієї з найбільш поширених вразливостей - *DOM* (*Document Object Model*) *XSS* вразливості). Проте представлені моделі не враховують останні тенденції *XSS* вразливості, а саме відмінність їх типів ("*stored XSS*", "*reflected XSS*" і *DOM Based XSS*) і необхідність їх виявлення.

Саме тому особливо актуальним завданням в цьому напрямі видається моделювання алгоритму виявлення *DOM* (*Document Object Model*) *XSS* вразливості з урахуванням комплексу трьох їх можливих типів.

Проведені дослідження показали, що вразливість *DOM XSS* є підвидом *XSS*, у разі якої результат атаки знаходиться не у відповіді сервера і, відповідно, не в *HTML*-коді, а в *DOM*-структурі *HTML*-сторінки. При цьому в режимі "*stored XSS*" здійснюється передача і зберігання *XSS* на сервері. Надалі на цю сторінку перенаправляються користувачі.

У режимі "*reflected XSS*" повертається в тілі відповіді від сервера на конкретний запит з самою *XSS*. Результати атак за допомогою таких вразливостей можна виявити лише в процесі виконання або аналізі *DOM*-структури. Сам механізм атаки, а саме ін'єкція *JavaScript* коду в уразливий сегмент, залишається незмінним.

Слід зауважити, що одним з найменш математично формалізованих і досліджуваних типів *XSS* є *DOM Based XSS*. Можливо, це пов'язано з тим, що

навіть сучасними сканерами їх не часто можна виявити і, відповідно, представити чіткий алгоритм виконання операцій аналізу вразливості.

Для математичної формалізації алгоритму виявлення комплексу *DOM XSS* вразливостей різних типів скористаємося основними положеннями мережевого *GERT*-моделювання, детально описаними в роботах [3, 20, 25-28].

Графічне представлення алгоритму виявлення комплексу *DOM XSS* вразливостей наведено на рис. 4.1.

Відповідно до цього алгоритму, основні етапи можна описати таким чином:

1. З коду аналізованої сторінки витягаються усі теги <*script*> і формується список тегів для аналізу.

2. Виконується аналіз вмісту тега. При цьому, якщо теги не містять код, а посилаються на віддалений файл, виконується звернення до файлу і отримання коду з нього. У вмісті файлу знаходяться потенційно небезпечні ділянки коду (*sink*), які використовують вхідні дані клієнта (*source*).

Проведений аналіз і дослідження [2, 3, 21-24] показали, що прикладами джерел можуть бути наступні:

Запис в *HTML*-код сторінки:

```
document.write(…)
document.writeln(…)
document.body.innerHtml=…
```

Зміна *DOM* безпосередньо (включаючи події *DHTML*):

```
document.forms[0].action=… (та інші варіації)
document.attachEvent(…)
document.create…(…)
document.execCommand(…)
document.body. … (доступ до DOM через об’єкт body)
window.attachEvent(…)
```

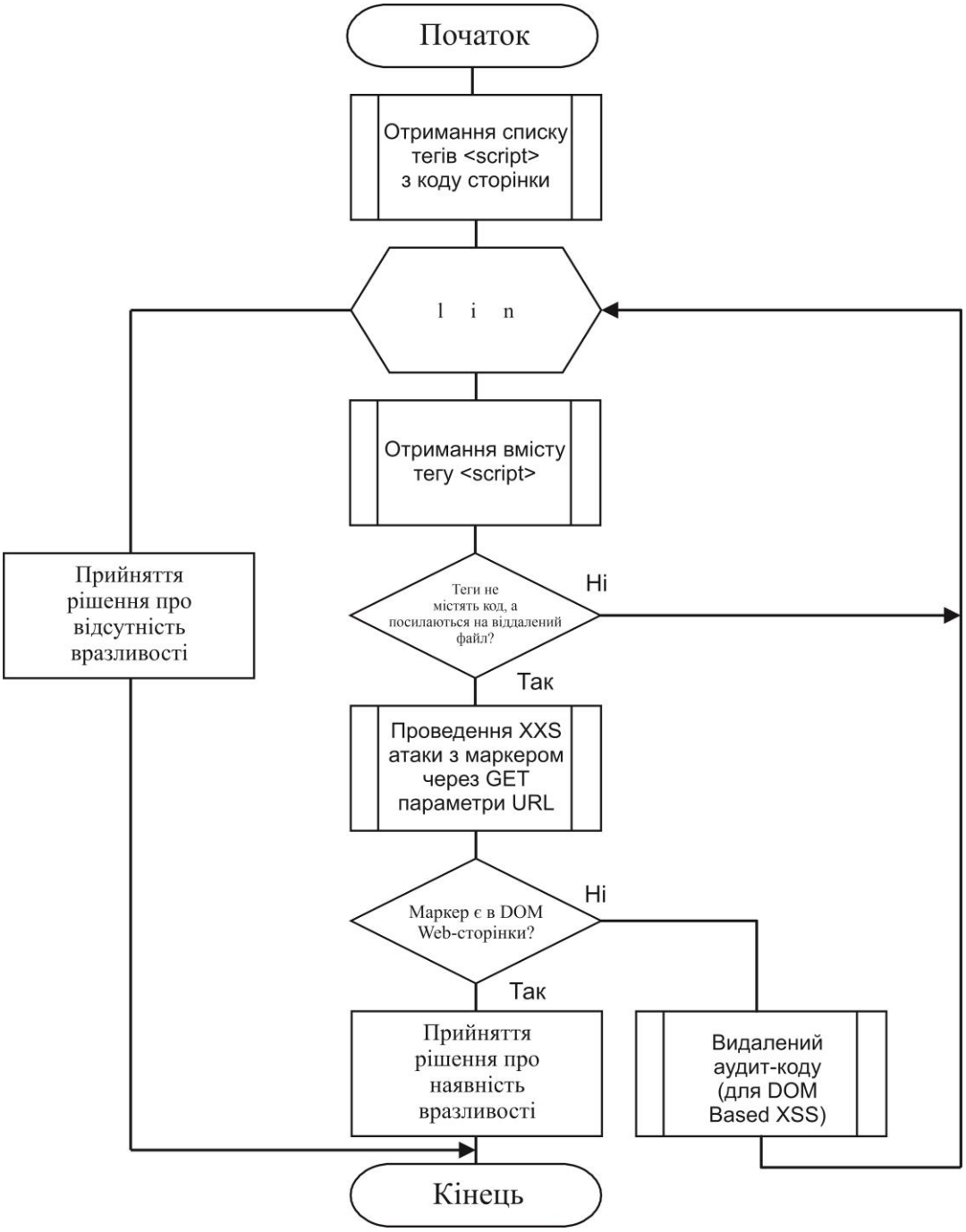


Рис. 4.1. Структурна схема алгоритму виявлення комплексу *DOM XSS* вразливостей

Зміна *URL* документу:

`document.location=…` (а також присвоювання значень *href*, *host* і *hostname* об'єкта location)

`document.location.hostname=…`

`document.location.replace(…)`

```
document.location.assign(…)
document.URL=…
window.navigate(…)
```

Відкриття/модифікація об’єкта *window*:

```
document.open(…)
window.open(…)
```

`window.location.href=…` (а також присвоювання значень *href*, *host* і *hostname* об’єкта *location*).

Виконання скрипту безпосередньо:

```
eval(…)
window.execScript(…)
window.setInterval(…)
window.setTimeout(…)
```

Приклади *sink*:

```
document.write
(element).innerHTML
eval
setTimout / setInterval
execScript
```

Даний перелік далеко не є вичерпним, і в основному є працею експертів лабораторії SecurityLab [29].

3. Якщо в коді тега використовується *source*, виконується атака з певним маркером, який можна відстежити в *DOM* структурі сторінки після виконання коду (наприклад, ін'єкція певного текстового вмісту в *DOM*).

4. Виконується перевірка вмісту *DOM*. Якщо в результаті атаки маркер знаходиться в *DOM*, можна зробити висновок про наявність *DOM* вразливості.

5. Після впровадження даних вручну і аналізу результатів, виконаних на перших 4 етапах, виконується аудит коду (може бути здійснений віддалено).

6. Кроки 2-5 виконуються для кожного тега *script* на сторінці.

Для побудування формальної моделі алгоритму виявлення комплексу *DOM XSS* вразливостей обрано стохастичну *GERT*-мережу.

Проведені дослідження показали, що *GERT* (*Graphical Evaluation and Review Technique*) є методом вивчення і аналізу стохастичних мереж, використовуваних для опису логічного взаємозв'язку між частинами проекту або етапами процесу [3, 20, 25-28].

Головною метою *GERT* є оцінка логіки мережі та тривалості активності і отримання висновку про необхідність виконання деяких активностей.

Мережі *GERT* складаються з вузлів типу *AND*, *INCLUSIVE-OR* і *EXCLUSIVE-OR*, а також гілок з двома і більше параметрами. Гілка має напрям, має вузол початку і вузол кінця. Параметри гілки містять:

1) ймовірність проходження гілки ($P_a$) за умови, що вузол, який є джерелом гілки, був реалізований;

2) час ($t_a$) проходження гілки, якщо вона буде реалізована.

Час $t_a$ може бути випадковою величиною. Якщо гілка не є частиною реалізації мережі, тобто під час виконання процесу активність, пов'язана з гілкою, не відбувається, то $t_a = 0$.

Вузол в стохастичній мережі *GERT* складається з функції входу (контрібутивної функції) і функції виходу (дистрібутивної функції). Кожна з функцій описується певним логічним відношенням відносно пов'язаних гілок.

В цілому, проведені дослідження показали, що *GERT*-моделювання є ефективним способом визначення заздалегідь невідомих законів і функцій розподілу випадкових величин при відомому алгоритмі функціонування (процесу).

Саме тому, в якості інструменту математичного моделювання нами було обрано *GERT*-моделювання.

### 4.1.2 GERT-модель технології тестування комплексу *DOM XSS* вразливостей

Побудуємо відповідно до представленого опису мережеву *GERT*-модель технології тестування комплексу *DOM XSS* вразливостей. Графічне зображення *GERT*-моделі представлене на рис. 4.2.

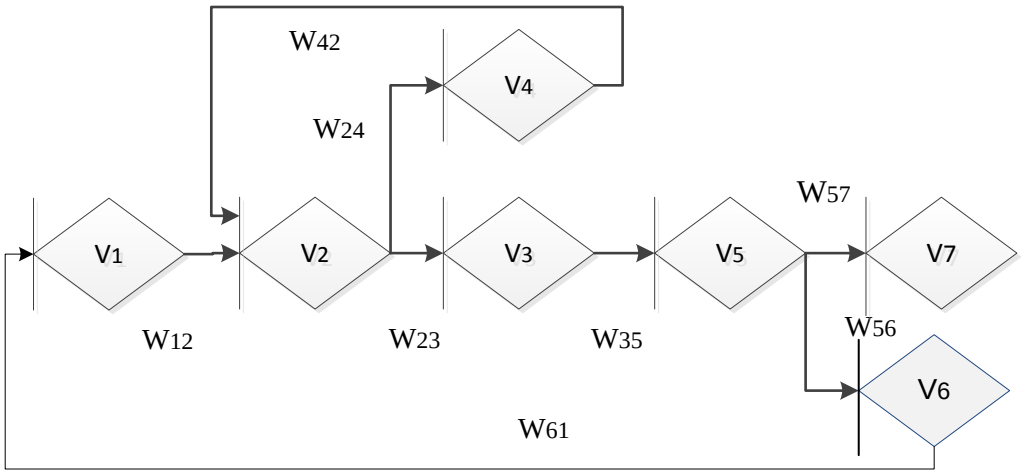


Рис. 4.2. GERT-модель технології тестування комплексу *DOM XSS* вразливостей

У представленій мережі вузли графа інтерпретуються станами комп'ютерної системи в процесі функціонування *DOM* структури, а гілки графа - ймовірнісно-часовими характеристиками переходів між станами. Зокрема, гілка (1,2) описує процес отримання і аналізу вмісту тега. Гілка (2,3) відображає процес виконання атаки у разі наявності "*source*" структури. Гілка (2,4) визначається процедурами звернення до вмісту віддаленого файлу (пошук "*sink*"). Гілка (4,2) характеризує повернення на виконання атаки. Гілка (3,5) описує продовження атаки, зокрема перевірку вмісту *DOM*. Гілка (5,6) характеризує одну з основних особливостей аналізу алгоритму *XSS* вразливостей різних типів - автоматичний аудит коду (при необхідності

віддалений). Гілка (6,1) відображає процес переходу до нового тега. Далі гілка (5,7) характеризує завершальну стадію прийняття рішення про вразливість.

Характеристики гілок моделі представлені в табл. 4.1.

Еквівалентна *W*-функція часу виконання алгоритму тестування комплексу *DOM XSS* різних типів (у тому числі *DOM Based XSS*) вразливостей дорівнює:

$$W_E(s) = \frac{W_{12}W_{23}W_{35}W_{56} + W_{12}W_{24}W_{42}W_{23}W_{35}W_{57}}{1 - W_{12}W_{23}W_{35}W_{56}W_{61} - W_{12}W_{24}W_{42}W_{23}W_{35}W_{56}W_{61}} =$$

$$= \frac{p_1 p_2^2 \lambda_1 \lambda_2^2 \left( p_4 \lambda_4 (\lambda_3 - s)^2 (\lambda_5 - s) + p_3^2 q_1 \lambda_3^2 \lambda_5 (\lambda_4 - s) \right)}{(\lambda_4 - s) \begin{pmatrix} (\lambda_1 - s)(\lambda_2 - s)^2 (\lambda_3 - s)^2 (\lambda_5 - s) - \\ - p_1 \lambda_1 p_2^2 \lambda_2^2 q_1 \lambda_5 (\lambda_3 - s)^2 - p_1 p_2^2 p_3^2 p_4 \lambda_1 \lambda_2^2 q_1 \lambda_3^2 \lambda_4 \lambda_5 \end{pmatrix}} \qquad (4.1)$$

де $1 - p_4 = q_1$.

Особливість даного процесу полягає в різнорідності аналізованих і оброблюваних даних. При цьому можливі різні випадки організації зворотного зв'язку. На рис. 4.2 ці цикли зафіксовані у вигляді переходів $W_{12} \to W_{24} \to W_{42}$, $W_{12} \to W_{23} \to W_{35} \to W_{56} \to W_{61}$.

Таблиця 4.1. Характеристики гілок моделі

| № з/п | Гілка | *W*-функція | Ймовірність | Твірна функція моментів |
|---|---|---|---|---|
| 1 | (1,2) | $W_{12}$ | *p1* | $\lambda_1/(\lambda_1 - s)$ |
| 2 | (2,3) | $W_{23}$ | *p2* | $\lambda_2/(\lambda_2 - s)$ |
| 3 | (2,4) | $W_{24}$ | *p3* | $\lambda_3/(\lambda_3 - s)$ |
| 4 | (3,5) | $W_{35}$ | *p2* | $\lambda_2/(\lambda_2 - s)$ |
| 5 | (5,6) | $W_{56}$ | *p4* | $\lambda_4/(\lambda_4 - s)$ |
| 6 | (6,1) | $W_{61}$ | *1 – p4* | $\lambda_5/(\lambda_5 - s)$ |
| 7 | (4,2) | $W_{42}$ | *p3* | $\lambda_3/(\lambda_3 - s)$ |
| 8 | (5,7) | $W_{57}$ | *p4* | $\lambda_4/(\lambda_4 - s)$ |

Для *GERT*-мереж з циклами не існує простих методів знаходження особливих точок функції $\Phi_E(z)$ заміни дійсних змінних $(z=-is)$, де s – дійсна змінна.

Це пояснюється тим, що для того щоб визначити особливі точки потрібно знайти рішення нелінійних рівнянь, до того ж рівень складності початкового рівняння прямо пропорційно залежить від складності структури *GERT*-мережі. Саме тому запропоновано зробити відповідну заміну в ході моделювання.

Виконуючи комплексне перетворення $z=-is$, отримаємо:

$$\Phi(z)=\frac{uz^3+vz^2+bz+k}{(\lambda_4+z)(z^6+cz^5+dz^4+gz^3+hz^2+wz+m)}, \qquad (4.2)$$

де $u=-p_1p_2^2p_4\lambda_1\lambda_2^2\lambda_4$,

$v=p_1p_2^2p_4\lambda_1\lambda_2^2\lambda_4(\lambda_5+2\lambda_3)$,

$b=-p_1p_2^2p_4\lambda_1\lambda_2^2\lambda_4\lambda_3(2\lambda_5-\lambda_3)$,

$k=-p_1p_2^2\lambda_1\lambda_2^2\lambda_3^2\lambda_4\lambda_5(p_4+p_3^2q_1)$,

$c=\lambda_1+2\lambda_2+2\lambda_3+\lambda_4+\lambda_5$,

$d=-(2\lambda_3\lambda_5\lambda_4+\lambda_1\lambda_5\lambda_4+2\lambda_2\lambda_5\lambda_4+\lambda_3^2+2\lambda_1\lambda_3+4\lambda_2\lambda_3+2\lambda_1\lambda_2+\lambda_2^2)$,

$$g=\begin{pmatrix}\lambda_3^2\lambda_4\lambda_5+4\lambda_1\lambda_2\lambda_4\lambda_5+4\lambda_2\lambda_3\lambda_4\lambda_5+\lambda_2^2+\lambda_3^2\lambda_1+2\lambda_3^2\lambda_2+4\lambda_1\lambda_2\lambda_3\lambda_4+\\+2\lambda_2^2\lambda_3+\lambda_2^2\lambda_1+\lambda_3^2\lambda_4+\lambda_2^2\lambda_4\end{pmatrix},$$

$$h=-\begin{pmatrix}\lambda_1\lambda_3^2\lambda_4\lambda_5+2\lambda_2\lambda_3^2\lambda_4\lambda_5+4\lambda_1\lambda_2\lambda_3\lambda_4\lambda_5+2\lambda_2^2\lambda_3\lambda_4\lambda_5+\lambda_2^2\lambda_3^2\lambda_4+\\+2\lambda_1\lambda_2^2\lambda_3\lambda_4-p_1p_2^2p\ q_1\lambda_1\lambda_2\lambda_4\lambda_5\end{pmatrix},$$

$w=\lambda_1\lambda_2\lambda_3^2\lambda_4\lambda_5+\lambda_2^2\lambda_3^2\lambda_4\lambda_5+2\lambda_1\lambda_2^2\lambda_3\lambda_4\lambda_5+\lambda_1\lambda_2^2\lambda_4\lambda_3-2p_1p_2^2p_4q_1\lambda_1\lambda_2\lambda_3\lambda_4\lambda_5$,

$m=p_1p_2^2p_4q_1\lambda_1\lambda_2\lambda_3^2\lambda_4\lambda_5+p_1p_2^2p_3p_4q_1\lambda_1\lambda_2^2\lambda_3^2\lambda_4\lambda_5-\lambda_1\lambda_2^2\lambda_3\lambda_4\lambda_5$.

Щільність розподілу ймовірностей часу виконання алгоритму аналізу *DOM XSS* вразливості:

$$\varphi(x)=\frac{1}{2\pi i}\int\limits_{-i\infty}^{i\infty}e^{zx}\frac{\mathrm{uz}^3+vz^2+bz+j}{(\lambda_4+z)\left(z^6+cz^5+dz^4+gz^3+hz^2+wz+m\right)}dz, \qquad (4.3)$$

де операція інтегрування виконується за допомогою інтегралу Бромвіча-Вагнера [30].

Спосіб інтегрування залежить від того, чи має функція Φ(z) лише прості полюси, або полюси деякого порядку. У тому випадку, коли функція Φ(z) має лише прості полюси, вираз $e^{zx}\Phi(z)$ можна представити у вигляді:

$$e^{zx}\Phi(\mathrm{z})=\frac{e^{zx}\left(\mathrm{uz}^3+vz^2+bz+j\right)}{z^7+\gamma_6z^6+\gamma_5z^5+\gamma_4z^4+\gamma_3z^3+\gamma_2z^2+\gamma_1z+\gamma_0}=\frac{\mu(z)}{\psi(z)}, \qquad (4.4)$$

де $\gamma_6=c+\lambda_4$, $\gamma_5=\lambda_4c+d$, $\gamma_4=\lambda_4d+g$, $\gamma_3=\lambda_4g+h$, $\gamma_2=\lambda_4h+w$, $\gamma_1=\lambda_4w+m$, $\gamma_0=\lambda_4m$.

Тоді щільність розподілу часу виконання алгоритму аналізу *DOM XSS* вразливості всіх типів дорівнює:

$$\varphi(x)=\sum_{k=1}^{7}\mathrm{Res}\left[e^{zx}\Phi(z)\right]=\sum_{k=1}^{7}\frac{\mu(z_k)}{\psi'(z_k)}=$$
$$=\sum_{k=1}^{7}\frac{e^{zx}\left(\mathrm{uz}^3+vz^2+bz+j\right)}{7z_k^6+6\gamma_6z_k^5+5\gamma_5z_k^4+4\gamma_4z_k^3+3\gamma_3z_k^2+2\gamma_2z_k+\gamma_1}. \qquad (4.5)$$

Функція $\Phi(z)$ полюсами, які визначаються коренями рівняння $z^7+\gamma_6z^6+\gamma_5z^5+\gamma_4z^4+\gamma_3z^3+\gamma_2z^2+\gamma_1z+\gamma_0=0$, може мати і полюс другого або більшого порядку. Тоді щільність розподілу часу передачі повідомлення $\varphi(x)$ знаходиться по формулі знаходження лишків $\mathrm{r}_{-1}$ від полюсів $z_k$ порядку $n$:

$$r_{-1}=\frac{1}{(n-1)!}\lim_{z\to z_k}\frac{d^{n-1}\left((z-z_k)^ne^{zx}\Phi(z)\right)}{dz^{n-1}}. \qquad (4.6)$$

Вираз (4.6) є дробово-раціональною функцією відносно $z$ зі степеню знаменника більшим, ніж степінь чисельника. Тому для нього виконуються умови леми Жордана [30, 31].

Многочлен $z^7 + \gamma_6 z^6 + \gamma_5 z^5 + \gamma_4 z^4 + \gamma_3 z^3 + \gamma_2 z^2 + \gamma_1 z + \gamma_0$ породжує сім полюсів. Рішення рівняння

$$z^7 + \gamma_6 z^6 + \gamma_5 z^5 + \gamma_4 z^4 + \gamma_3 z^3 + \gamma_2 z^2 + \gamma_1 z + \gamma_0 = 0 \qquad (4.7)$$

може бути знайдене будь-яким чисельним методом, наприклад за [31]. В результаті обчислюються особливі точки $z_1$, $z_2$, $z_3$, $z_4$, $z_5$, $z_6$, $z_7$.

Таким чином, на основі експоненційної *GERT*-мережі розроблено математичну модель технології тестування комплексу *DOM XSS* вразливостей усіх типів ("*stored XSS*", "*reflected XSS*" і *DOM Based XSS*), яка відрізняється від відомих урахуванням їх специфіки і необхідності автоматичного аудиту *DOM Based XSS* окремо.

Розроблена модель може бути використана для дослідження Інтернет *Web*-застосунків в мережевих структурах, а також при розробці нових засобів і протоколів захисту даних в комп'ютерних системах і мережах.

Застосування експоненційних стохастичних моделей *GERT* дасть змогу використовувати розраховані в аналітичному вигляді результати, а саме щільність розподілу та функції для дослідження і комплексного аналізу складніших комп'ютерних систем математичними методами.

### 4.1.3 Дослідження *GERT*-моделі технології тестування комплексу *DOM XSS* вразливостей

Розглянемо приклад *XSS* атаки через *DOM* (аналогічний алгоритм простого використання клієнтського скрипта для небезпечної переадресації браузеру до іншого ресурсу).

Знайдемо щільності розподілу $\varphi(x)$ ймовірностей часу виконання алгоритму за умови, що $z$ обираються як корені рівняння $z^7 + \gamma_6 z^6 + \gamma_5 z^5 + \gamma_4 z^4 + \gamma_3 z^3 + \gamma_2 z^2 + \gamma_1 z + \gamma_0 = 0$, умовні ймовірності

й інтенсивності в гілках *GERT*-мережі мають наступні значення: $p_1 = 0.999$, $p_2 = 0.6$, $p_3 = 0.4$, $p_4 = 0.99$, $\lambda_1 = 0.9999$, $\lambda_2 = 0.79$, $\lambda_3 = 0.29$, $\lambda_4 = 0.39$. Вказані значення визначено за спостереженнями процесів тестування вразливостей з перевіркою адекватності результатів моделювання.

З урахуванням наведених ознак *GERT*- мережі, відповідно до виразу (4.2), а також використовуючи спеціалізований математичний пакет *Mathcad*, отримаємо, що в знаменнику виразу (4.3) сформований поліном

$$z^6+4.07z^5+6.66z^4+6.529z^3+1.592z^2+0.617z+0.164= 0. \qquad (4.8)$$

Корені цього полінома (і відповідно функція $\Phi(z)$) дорівнюють $z_7=-\lambda_4$ як окремого множника до розкриття дужок з (4.3), та інші:

$z_1 \approx -5.50538139377208$, $(P(z_1) \approx 0$; iter = 1);

$z_2 \approx -0.0498463517249773 + i\ 0.331259064468874$,

$(P(z_2) \approx -0.000162 - i\ 0.00073$; iter = 3);

$z_3 \approx -0.0495665029547472 - i\ 0.331246931512067$,

$(P(z_3) \approx -0.000307 + i\ 0.00013$; iter = 3);

$z_4 \approx -0.28764249382953$,

$(P(z_4) \approx 0.000129$; iter = 3);

$z_5 \approx -0.623543971678568 - i\ 0.731454207007899$, $(P(z_5) \approx 0$; iter = 3);

$z_6 \approx -0.623607782943707 + i\ 0.731584150633824$, $(P(z_6) \approx 0$; iter = 2).

Дослідимо залежність функції Φ(z) від інтенсивності $\lambda_2$, що є одним з основних показників в даному алгоритмі (інтенсивність $\lambda_2$ характеризує виконання атаки у разі наявності «*source*» структури).

На рис. 4.3 представлена крива графіка залежності функції Φ(z) від інтенсивності $\lambda_2$ в розглянутих вище умовах. Як видно з рисунку 4.3, випадкова величина $\lambda_2$ розподілена відповідно до показникового закону.

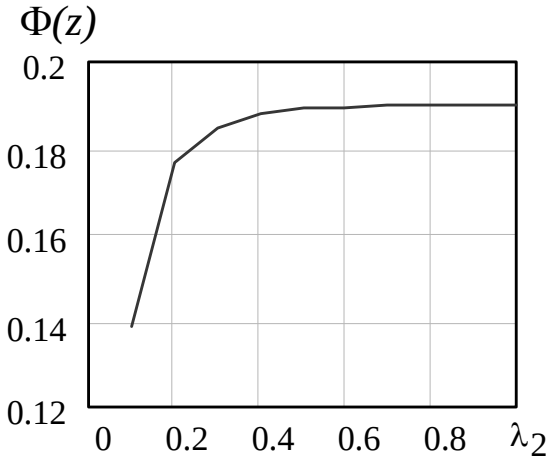


Рис. 4.3. Графік залежності функції Φ(z) від інтенсивності $\lambda_2$

Використовуючи отримані значення, знайдемо $\varphi(x)$. Відповідно до формули (4.5) та $z_k = a_k + \delta_k i$, $\varphi(x)$ дорівнює:

$$\varphi(x) = \sum_{k=1}^{7} Res_{z=z_k}\left[e^{zx}\Phi(z)\right] =$$
$$= \sum_{k=1}^{7} \frac{e^{(a_k+\delta_k i)x}\left(u(a_k+\delta_k i)^3 + v(a_k+\delta_k i)^2 + b(a_k+\delta_k i) + j\right)}{\left(\begin{array}{l} 7(a_k+\delta_k i)^6 + 6\gamma_6(a_k+\delta_k i)^5 + 5\gamma_5(a_k+\delta_k i)^4 + 4\gamma_4(a_k+\delta_k i)^3 + 3\gamma_3(a_k+\delta_k i)^2 + \\ + 2\gamma_2(a_k+\delta_k i) + \gamma_1 \end{array}\right)} \qquad (4.9)$$

З [32] відомо, що сума значень будь-якої дробово-раціональної функції

$$f(z) = \frac{d_m z^m + d_{m-1} z^{m-1} + \ldots d_1 z + d_0}{\ell_m z^m + \ell_{m-1} z^{m-1} + \ldots + \ell_1 z + \ell_0}, \quad d_m \neq 0, \ell_m \neq 0,$$

досліджуваною при значеннях комплексних зв'язаних аргументів, може бути представлена у вигляді:

$$\frac{(\tau + i\beta)}{(\chi + i\beta)} + \frac{(\tau - i\theta)}{(\chi - i\theta)},$$

де $\tau, \beta, \chi, \theta$ - деякі коефіцієнти.

Використовуючи вираз Ейлера [30], отримаємо:

$$\varphi(x) = \sum_{k=1}^{7} \mathrm{Res}_{z=z_k}\left(e^{zx}\Phi(z)\right) = \sum_{k=1}^{7}\left(e^{(a_k+\delta_k i)x}\frac{\tau_k + i\beta_k}{\chi_k + i\theta_k} + e^{(a_k-\delta_k i)x}\frac{\tau_k - i\beta_k}{\chi_k - i\theta_k}\right) =$$
$$= \sum_{k=1}^{7} \frac{2e^{a_k x}}{\chi^2 + \theta^2}\left((\tau_k\chi_k + \beta_k\theta_k)\cos(\delta_k x) + (\tau_k\chi_k - \beta_k\theta_k)\sin(\delta_k x)\right), \qquad (4.10)$$

де $\tau = a^3u - 3a\delta^2u + a^2v - \delta^2v + a\gamma + \chi$,

$\beta = 3a^2\delta u - \delta^3u + 2abv + \delta b$,

$\chi = 7a^6 - 10a^4\delta^2 + 105a^2\delta^4 - 7\delta^6 + 6\gamma_6a^5 - 60\gamma_6a^3\delta^2 + 30\gamma_6a\delta^4 + 5\gamma_5a^4 -$
$- 30\gamma_5a^2\delta^2 + 5\gamma_5\delta^4 + 4\gamma_4a^3 - 12\gamma_4a\delta^2 + 3\gamma_3a^2 - 3\gamma_3\delta^2 + 2\gamma_2a + \gamma_1,$

$\theta = 49a^5\delta - 140a^3\delta^3 + 49a\delta^5 + 30\gamma_6a^4\delta - 60\gamma_6a^2\delta^3 + 6\gamma_6\delta^5 + 20\gamma_5a^3\delta -$
$- 20\gamma_5a\delta^3 + 12\gamma_4a^3\delta - 4\gamma_4\delta^3 + 6\gamma_3a\delta + 2\gamma_2\delta.$

На рис. 4.4 представлені крива щільності розподілу $\varphi(x)$ ймовірності часу виконання алгоритму виявлення комплексу *DOM XSS* вразливостей для приведених вище умов (як вхідні дані використовувалися корені полінома (4.8)).

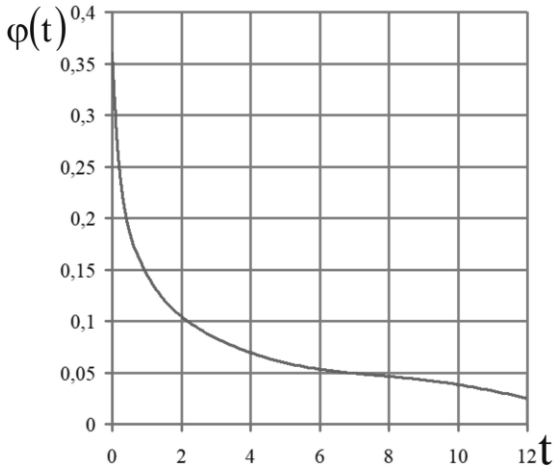


Рис. 4.4. Графік щільності розподілу $\varphi(t)$ ймовірності часу виконання алгоритму аналізу *DOM XSS* вразливості

Зовнішній вигляд графіку дає підстави припустити, що випадкова величина часу виконання алгоритму аналізу *DOM XSS* вразливості має гамма-розподіл.

Перевіримо цю гіпотезу за критерієм $\chi^2$ Пірсона [33]:

$$\chi^2 = N^* \sum_{i=1}^{k} (P_i^* - P_i)^2 / P_i ,$$

де $k$ - число розрядів (інтервалів) статистичного ряду; $P_i^*$ і $P_i$ - "статистична" і теоретична ймовірності події.

В результаті експерименту було отримано теоретичні значення $\chi^2$ і табличне значення $\overline{\chi^2}$, зворотнє правосторонній ймовірності розподілу $\chi^2$.

Проведена перевірка показала, що висунену гіпотезу можна вважати правдоподібною або, принаймні, такою, що не суперечить отриманим при математичному моделюванні результатам.

## 4.2 Математична модель технології тестування вразливості до *SQL*-ін'єкцій

### 4.2.1 Алгоритм аналізу вразливості до *SQL*-ін'єкцій

Проведені дослідження [12, 34] показали, що на основі аналізу методології тестування вразливості *Web*-застосунків до *DOM XSS* і матеріалів *Open Web Application Security Project* [12, 34-36], можна розробити алгоритм аналізу вразливості *Web*-застосунків до *SQL*-ін'єкцій.

Відмінною особливістю цього алгоритму є врахування лише вразливості, яка є в *GET*-параметрах *URL* і використовує лише сліпий метод ін'єкції *SQL*-коду, що використовує особливість використання булевих операторів в *SQL*-запитах (*Boolean blind SQL injection*) [2, 3].

Структурна схема алгоритму аналізу вразливості *Web*-застосунку до *SQL*-ін'єкцій представлена на рис. 4.5.

Відповідно до представленого алгоритму, його етапи можна описати наступним чином:

1. З введеного *URL*-посилання виходить список *GET*-параметрів.

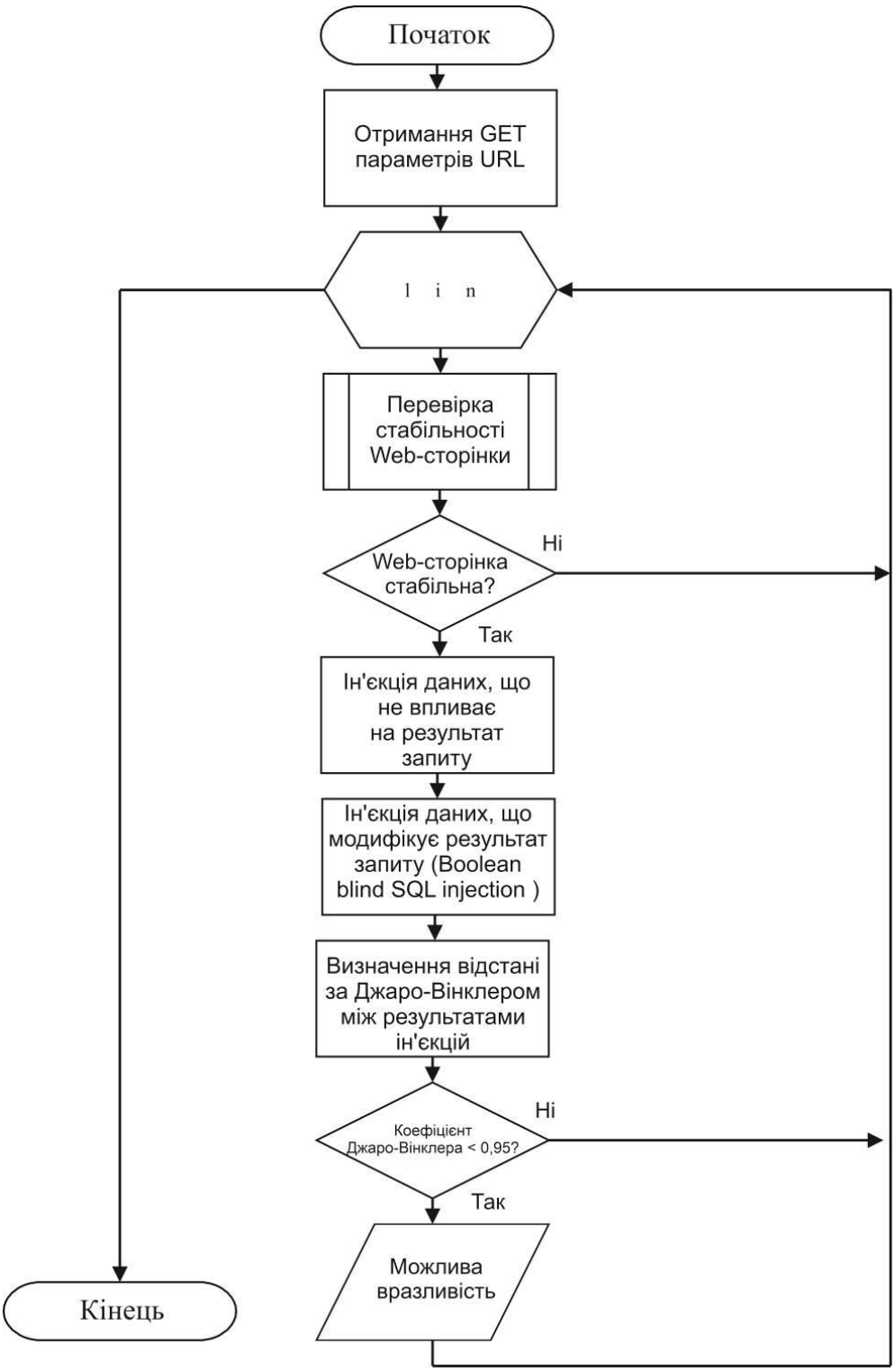


Рис. 4.5. Структурна схема алгоритму аналізу вразливості *Web*-застосунку до *SQL*-ін'єкцій

2. Виконується перевірка стабільності *Web*-сторінки. Для цього виконується два послідовні запити до *Web*- сторінки і обчислюється відстань

між змістом *HTML*-коду сторінки за допомогою критерію Джаро-Вінклера [37]. Якщо значення критерію менше певного порогового значення, виконувати подальший аналіз неможливо.

3. У параметр *GET* запиту виконується ін'єкція *SQL*-коду, який не змінює результат запиту до бази даних, і зберігається результуючий *HTML*-код.

4. У параметрі *GET* запиту виконується ін'єкція *SQL*-коду, який змінює результат запиту до бази даних, приводить або до отримання повного набору даних з таблиці, або до відсутності результату, після чого зберігається результуючий *HTML*-код.

5. За допомогою критерію Джаро-Вінклера виконується порівняння результатів ін'єкції *SQL*-коду. Якщо значення критерію менше певного порогового значення, то в цьому *GET*-параметрі є можлива вразливість до *SQL*-ін'єкції.

6. Кроки 2-5 повторюються для усіх параметрів *GET* запиту наданого *URL*.

На основі представленого алгоритму розробимо *GERT*-модель технології тестування вразливості до *SQL*-ін'єкцій.

### 4.2.2 *GERT*-модель технології тестування вразливості до *SQL*-ін'єкцій

Побудуємо відповідно до представленого опису мережеву *GERT*-модель технології тестування вразливості до *SQL*-ін'єкцій. Графічне зображення *GERT*-моделі представлене на рис. 4.6.

У представленій мережі вузли графа інтерпретуються станами комп'ютерної системи в процесі тестування вразливості до *SQL*-ін'єкцій, а

гілки графа - ймовірнісно-часовими характеристиками переходів між станами.

Зокрема, гілка (1,2) характеризує час отримання і аналізу *GET*-параметрів з введеного *URL*-посилання.

Гілка (2,3) відображає час відправлення первинних і вторинних запитів до *Web*-сторінки.

Гілка (3,4) задає випадковий час порівняння сторінок (час обчислення відстані між змістом *HTML*-коду сторінки за допомогою критерію Джаро-Вінклера).

Гілка (4,5) характеризує час, за який виконується ін'єкція *SQL*-коду, який не змінює результат запиту до бази даних, а також який змінює результат запиту до бази даних відповідно.

Гілка (5,6) характеризує час порівняння результатів ін'єкції *SQL*-коду.

Гілка (4,2) характеризує часові характеристики повернення системи в первинний стан, коли значення критерію Джаро-Вінклера менше певного порогового значення, в той же час гілка (6,2) відображає часові характеристики переходу до нової перевірки у випадку, якщо значення критерію Джаро-Вінклера більше певного порогового значення.

Характеристики гілок моделі представлені в табл. 4.2.

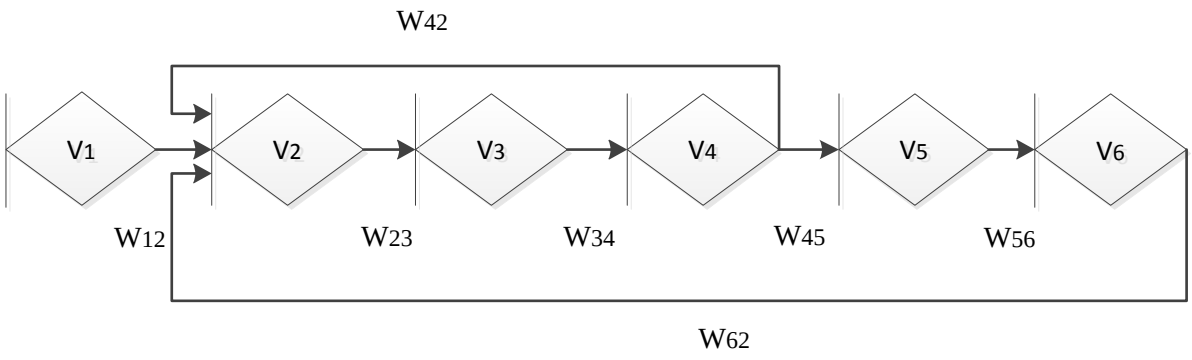


Рис. 4.6. *GERT*-модель технології тестування вразливості до *SQL*-ін'єкцій

Таблиця 4.2. Характеристики гілок моделі технології тестування вразливості до *SQL*-ін'єкцій

| № з/п | Гілка | *W*-функція | Ймовірність | Твірна функція моментів |
|---|---|---|---|---|
| 1 | (1,2) | $W_{12}$ | $p_1$ | $\lambda_1/(\lambda_1 - s)$ |
| 2 | (2,3) | $W_{23}$ | $p_2$ | $\lambda_2/(\lambda_2 - s)$ |
| 3 | (3,4) | $W_{34}$ | $p_3$ | $\lambda_3/(\lambda_3 - s)$ |
| 4 | (4,5) | $W_{45}$ | $p_4$ | $\lambda_4/(\lambda_4 - s)$ |
| 5 | (5,6) | $W_{56}$ | $p_5$ | $\lambda_3/(\lambda_3 - s)$ |
| 6 | (4,2) | $W_{42}$ | $1 - p_4$ | $\lambda_5/(\lambda_5 - s)$ |
| 7 | (6,2) | $W_{62}$ | $p_6$ | $\lambda_6/(\lambda_6 - s)$ |

Еквівалентна *W*-функція часу виконання технології тестування вразливості до *SQL*-ін'єкцій дорівнює:

$$W_E(s) = \frac{W_{12}W_{23}W_{34}W_{45}W_{56}}{1 - W_{12}W_{23}W_{34}W_{42} - W_{12}W_{23}W_{34}W_{45}W_{56}W_{62}} =$$

$$= \frac{p_1 p_2 p_3 p_4 p_5 \lambda_1 \lambda_2 \lambda_3^2 \lambda_4 (\lambda_3 - s)(\lambda_5 - s)(\lambda_6 - s)}{\left[\begin{array}{l}(\lambda_1 - s)(\lambda_2 - s)(\lambda_3 - s)^2(\lambda_4 - s)(\lambda_5 - s)(\lambda_6 - s) - \\ -\left(\begin{array}{l} p_1 p_2 p_3 \lambda_1 \lambda_2 \lambda_3 \times \\ \times\left(q_1 \lambda_5 \left(\lambda_3 \lambda_4 - \lambda_4 s - \lambda_3 s - s^2\right)(\lambda_6 - s) - p_4 p_5 p_6 \lambda_3 \lambda_4 \lambda_6 (\lambda_5 - s)\right)\end{array}\right)\end{array}\right]} \qquad (4.11)$$

де $1 - p_4 = q_1$.

Аналогічно підпункту 4.1.1., виконуючи комплексне перетворення $z = -is$, отримаємо

$$\Phi(z) = \frac{vz^2 + bz + k}{\left(z^7 + rz^6 + cz^5 + dz^4 + gz^3 + hz^2 + wz + m\right)} \qquad (4.12)$$

де $v = -p_1 p_2 p_3 p_4 p_5 \lambda_1 \lambda_2 \lambda_3^2 \lambda_4$,

$$b = p_1 p_2 p_3 p_4 p_5 \lambda_1 \lambda_2 \lambda_3^2 \lambda_4 (\lambda_5 + \lambda_6),$$

$$k = -p_1 p_2 p_3 p_4 p_5 \lambda_1 \lambda_2 \lambda_3^2 \lambda_4 \lambda_5 \lambda_6,$$

$$r = \lambda_1 + \lambda_2 + \lambda_4 + \lambda_5 - 2\lambda_3 - \lambda_6,$$

$$c = \begin{pmatrix} \lambda_1\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_5 + \lambda_2\lambda_5 + \lambda_3^2 + 2\lambda_3\lambda_6 - \lambda_4\lambda_6 - \lambda_5\lambda_6 - \lambda_1\lambda_6 - \lambda_4\lambda_5 - \\ -2\lambda_3\lambda_4 - 2\lambda_3\lambda_5 - \lambda_1\lambda_2 - 2\lambda_1\lambda_3 - 2\lambda_2\lambda_3 \end{pmatrix},$$

$$d = -\lambda_1\lambda_2\lambda_3\lambda_4\lambda_5\lambda_6 \begin{pmatrix} \frac{1}{\lambda_2\lambda_3\lambda_5} + \frac{1}{\lambda_1\lambda_3\lambda_5} + \frac{1}{\lambda_2\lambda_3\lambda_4} + \frac{1}{\lambda_1\lambda_3\lambda_4} + \frac{\lambda_3}{\lambda_1\lambda_2\lambda_4\lambda_5} + \\ + \frac{1}{\lambda_2\lambda_3\lambda_6} + \frac{1}{\lambda_1\lambda_3\lambda_6} + \frac{1}{\lambda_3\lambda_5\lambda_6} + \frac{2}{\lambda_2\lambda_5\lambda_6} + \frac{2}{\lambda_1\lambda_5\lambda_6} + \frac{1}{\lambda_3\lambda_4\lambda_6} + \\ + \frac{2}{\lambda_2\lambda_4\lambda_6} + \frac{2}{\lambda_1\lambda_4\lambda_6} - \frac{1}{\lambda_1\lambda_2\lambda_3} - \frac{2}{\lambda_1\lambda_2\lambda_5} - \frac{2}{\lambda_1\lambda_2\lambda_4} - \frac{1}{\lambda_3\lambda_4\lambda_5} - \\ - \frac{2}{\lambda_2\lambda_4\lambda_5} - \frac{2}{\lambda_1\lambda_2\lambda_6} - \frac{\lambda_3}{\lambda_1\lambda_2\lambda_5\lambda_6} - \frac{\lambda_3}{\lambda_1\lambda_2\lambda_4\lambda_6} - \frac{2}{\lambda_4\lambda_5\lambda_6} - \\ - \frac{\lambda_3}{\lambda_2\lambda_4\lambda_5\lambda_6} - \frac{\lambda_3}{\lambda_1\lambda_4\lambda_5\lambda_6} \end{pmatrix},$$

$$g = \left[ \left( \lambda_1\lambda_2\lambda_3\lambda_4\lambda_5\lambda_6 \begin{pmatrix} \frac{2}{\lambda_1\lambda_2} + \frac{\lambda_3}{\lambda_1\lambda_2\lambda_5} + \frac{\lambda_3}{\lambda_1\lambda_2\lambda_4} + \frac{2}{\lambda_4\lambda_5} + \\ + \frac{\lambda_3}{\lambda_2\lambda_4\lambda_5} + \frac{\lambda_3}{\lambda_1\lambda_4\lambda_5} + \frac{\lambda_3}{\lambda_1\lambda_2\lambda_6} - \\ - \frac{1}{\lambda_2\lambda_3} - \frac{1}{\lambda_1\lambda_3} - \frac{1}{\lambda_3\lambda_5} - \frac{2}{\lambda_2\lambda_5} - \\ - \frac{2}{\lambda_1\lambda_5} - \frac{1}{\lambda_3\lambda_4} - \frac{2}{\lambda_2\lambda_4} - \frac{2}{\lambda_1\lambda_4} - \\ - \frac{1}{\lambda_3\lambda_6} - \frac{2}{\lambda_2\lambda_6} - \frac{2}{\lambda_1\lambda_6} - \frac{2}{\lambda_5\lambda_6} - \\ - \frac{\lambda_3}{\lambda_2\lambda_5\lambda_6} - \frac{\lambda_3}{\lambda_1\lambda_5\lambda_6} - \frac{2}{\lambda_4\lambda_6} - \\ - \frac{\lambda_3}{\lambda_2\lambda_4\lambda_6} - \frac{\lambda_3}{\lambda_1\lambda_4\lambda_6} - \frac{\lambda_3}{\lambda_4\lambda_5\lambda_6} \end{pmatrix} \right) + p_1 p_2 p_3 q_1 \lambda_1\lambda_2\lambda_3\lambda_5 \right],$$

$$h=-\left[\left(\begin{array}{l}\lambda_1\lambda_2\lambda_3\lambda_4\lambda_5\lambda_6\times\\ \times\left(\begin{array}{l}\frac{\lambda_3}{\lambda_1\lambda_2}+\frac{\lambda_3}{\lambda_4\lambda_5}-\frac{1}{\lambda_3}-\\ -\frac{2}{\lambda_2}-\frac{2}{\lambda_1}-\frac{2}{\lambda_5}-\frac{\lambda_3}{\lambda_2\lambda_5}-\\ -\frac{\lambda_3}{\lambda_1\lambda_5}-\frac{2}{\lambda_4}-\frac{\lambda_3}{\lambda_2\lambda_4}-\\ -\frac{\lambda_3}{\lambda_1\lambda_4}-\frac{2}{\lambda_6}-\frac{\lambda_3}{\lambda_2\lambda_6}-\\ -\frac{\lambda_3}{\lambda_1\lambda_6}-\frac{\lambda_3}{\lambda_5\lambda_6}-\frac{\lambda_3}{\lambda_4\lambda_6}\end{array}\right)\end{array}\right)+\left(\begin{array}{l}p_1p_2p_3q_1\lambda_1\lambda_2\lambda_3\lambda_5\lambda_6\times\\ \times\left(\frac{1}{\lambda_5\lambda_6}+\frac{\lambda_3}{\lambda_4\lambda_5\lambda_6}-\frac{1}{\lambda_4\lambda_5}\right)\end{array}\right)\right],$$

$$w=\left[\left(\begin{array}{l}\lambda_1\lambda_2\lambda_3\lambda_4\lambda_5\lambda_6\times\\ \times\left(\begin{array}{l}2+\frac{\lambda_3}{\lambda_2}+\frac{\lambda_3}{\lambda_1}+\\ +\frac{\lambda_3}{\lambda_5}+\frac{\lambda_3}{\lambda_4}+\frac{\lambda_3}{\lambda_6}\end{array}\right)\end{array}\right)-\left(\begin{array}{l}p_1p_2p_3q_1\lambda_1\lambda_2\lambda_3\lambda_5\lambda_6\times\\ \times\left(-1-\frac{\lambda_3}{\lambda_4}-\frac{\lambda_3}{\lambda_6}\right)\end{array}\right)+p_4p_5p_6\lambda_3\lambda_4\lambda_6\right],$$

$$m=-\lambda_1\lambda_2\lambda_3^2\lambda_5\lambda_6\left(\lambda_4+p_1p_2p_3q_1-\frac{p_4p_5p_6}{\lambda_1\lambda_2}\right).$$

Аналогічно підпункту 4.1.1., щільність розподілу ймовірності часу виконання технології тестування вразливості до *SQL*-ін'єкцій дорівнює:

$$\varphi(x)=\frac{1}{2\pi i}\int_{-i\infty}^{i\infty}e^{zx}\frac{vz^2+bz+k}{\left(z^7+rz^6+cz^5+dz^4+gz^3+hz^2+wz+m\right)}dz, \qquad (4.13)$$

де операція інтегрування виконується за допомогою інтегралу Бромвіча-Вагнера [30].

Тоді вираз $e^{zx}\Phi(z)$ можна представити у вигляді:

$$e^{zx}\Phi(z)=\frac{e^{zx}\left(vz^2+bz+k\right)}{z^7+rz^6+cz^5+dz^4+gz^3+hz^2+wz+m}=\frac{\mu(z)}{\psi(z)}. \qquad (4.14)$$

Тоді щільність розподілу часу виконання алгоритму тестування вразливості до *SQL*-ін’єкцій дорівнює:

$$\varphi(x)=\sum_{k=1}^{7}Res\left[e^{zx}\Phi(z)\right]=\sum_{k=1}^{7}\frac{\mu(z_k)}{\psi(z_k)}=$$
$$=\sum_{k=1}^{7}\frac{e^{zx}\left(vz^2+bz+k\right)}{7z_k^6+6rz_k^5+5cz_k^4+4dz_k^3+3gz_k^2+2hz_k+w}. \quad (4.15)$$

Многочлен $z^7+rz^6+cz^5+dz^4+gz^3+hz^2+wz+m$ породжує сім полюсів. Рішення рівняння

$$z^7+rz^6+cz^5+dz^4+gz^3+hz^2+wz+m=0. \quad (4.16)$$

може бути знайдене будь-яким методом, наприклад, за формулами Вієта [30]. В результаті обчислюються особливі точки $z_1$, $z_2$, $z_3$, $z_4$, $z_5$, $z_6$, $z_7$.

Таким чином, на основі експоненційної *GERT*-мережі розроблено математичну модель технології тестування вразливості до *SQL*-ін'єкцій, яка відрізняється від відомих вдосконаленим способом визначення відстані між результатами ін'єкції. Використання в запропонованому способі критерію Джаро-Вінклера для порівняння результатів ін'єкції *SQL*-коду і введення порогового значення дозволить підвищити точність результатів тестування вразливості до *SQL*-ін'єкцій.

### 4.2.3 Дослідження *GERT*-моделі технології тестування вразливості до *SQL*-ін'єкцій

Розглянемо приклад атаки *SQL*-ін'єкцій. Суть таких ін'єкцій - впровадження в дані (передавані через *GET*, *POST* запити або значення *Cookie*) довільного *SQL*-коду. Знайдемо щільності розподілу $\varphi(x)$ ймовірностей часу виконання алгоритму за умови, що $z$ обираються як корені рівняння $\left(z^7+rz^6+cz^5+dz^4+gz^3+hz^2+wz+m\right)=0$, умовні ймовірності і інтенсивності в гілках *GERT*-мережі мають наступні значення: $p_1=p_2=p_3=p_4=p_5=0.999999$, $p_6=0.9$, $\lambda_1=\lambda_2=\lambda_3=0.9999$, $\lambda_4=0.8$, $\lambda_5=0.1$,

$\lambda_6 = 0.999999$. Значення є оцінкою на практичних вимірюваннях роботи системи аналізування на наявність атаки *SQL*-ін'єкцій.

З урахуванням наведених ознак *GERT*-мережі, відповідно до виразу (4.12), а також використовуючи спеціалізований математичний пакет *Mathcad*, отримаємо, що в знаменнику виразу (4.13) сформований поліном

$$z^7+0.1z^6+4.174z^5+2.471z^4+4.509z^3+4.128z^2+2.014z+0.169=0. \quad (4.17)$$

Корені цього поліному (і, відповідно, функція $\Phi(z)$) дорівнюють:

$z_1 \approx -2.11254039866286$, $(P(z_1) \approx 0;\ iter = 1)$;

$z_2 \approx -0.561885634027132$, $(P(z_2) \approx 0;\ iter = 4)$;

$z_3 \approx -0.208185977139001 - i\ 0.60944124336833$, $(P(z_3) \approx 0;\ iter = 5)$;

$z_4 \approx -0.208185883644938 + i\ 0.609441306673327$, $(P(z_4) \approx 0;\ iter = 4)$;

$z_5 \approx -0.103581224605665$, $(P(z_5) \approx 0;\ iter = 3)$;

$z_6 \approx -1.64718955898524 - i\ 0.775107663208$, $(P(z_6) \approx 0;\ iter = 1)$;

$z_7 \approx -1.64718955909435 + i\ 0.775107667929698$, $(P(z_7) \approx 0;\ iter = 4)$.

Дослідимо залежність функції $\Phi(z)$ від інтенсивності $z$.

На рис. 4.7 представлена крива графіка залежності функції $\Phi(z)$ від $z$ в умовах, що розглядаються вище. Як видно з рисунка, випадкова величина z розподілена відповідно до показового закону.

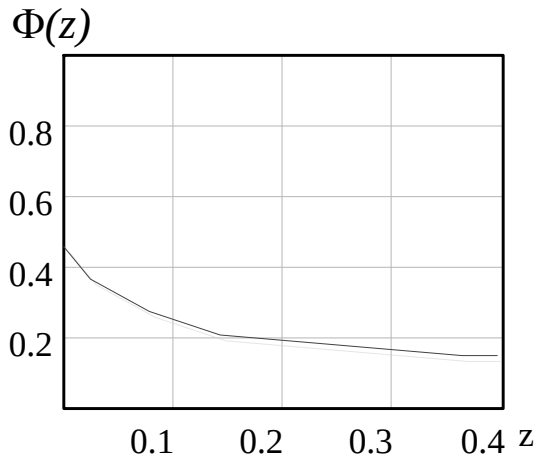


Рис. 4.7. Графік залежності функції $\Phi(z)$ від інтенсивності z

Аналогічно алгоритму розрахунку $\varphi(x)$, використаному в підпункті 4.1.2. знайдемо цю функцію і дослідимо її з використанням математичного пакету *Mathcad*. Відповідно до формули (4.15) $\varphi(x)$ дорівнює:

$$\varphi(x)=\sum_{k=1}^{7}\operatorname{Res}_{z=z_k}\left[e^{zx}\Phi(z)\right]=$$
$$=\sum_{k=1}^{7}\frac{e^{(a_k+\delta_k i)x}\left(v(a_k+\delta_k i)^2+b(a_k+\delta_k i)+k\right)}{\left(\begin{array}{l}7u(a_k+\delta_k i)^6+6r(a_k+\delta_k i)^5+5c(a_k+\delta_k i)^4+4d(a_k+\delta_k i)^3+3g(a_k+\delta_k i)^2+\\+2h(a_k+\delta_k i)+w\end{array}\right)} \qquad (4.18)$$

Аналогічно підходу, використаному в підпункті 4.1.2., використовуючи вирази Ейлера [33], отримаємо:

$$\varphi(x)=\sum_{k=1}^{7}\operatorname{Res}_{z=z_k}\left(e^{zx}\Phi(z)\right)=\sum_{k=1}^{7}e^{(a_k+\delta_k i)x}\frac{\tau+i\beta}{\gamma+i\theta}+e^{(a_k-\delta_k i)x}\frac{\tau-i\beta}{\gamma-i\theta}=$$
$$=\sum_{k=1}^{7}\frac{2e^{a_k x}}{\gamma^2+\theta^2}\left((\tau\gamma+\beta\theta)\cos(\delta_k x)+(\tau\gamma-\beta\theta)\sin(\delta_k x)\right), \qquad (4.19)$$

де $\tau=a^2v-\delta^2v+ab+k$,

$\beta=2a\delta v-\delta b$,

$\gamma=7ua^6-10ua^4\delta^2+105ua^2\delta^4-7u\delta^6+6ra^5-60ra^3\delta^2+30ra\delta^4+5ca^4-$
$-30ca^2\delta^2+5c\delta^4+4da^3-12da\delta^2+3ga^2-3g\delta^2+2ha+w$,

$\theta=49ua^5\delta-140ua^3\delta^3+49ua\delta^5+30ra^4\delta-60ra^2\delta^3+6r\delta^5+20ca^3\delta-$
$-20ca\delta^3+12da^3\delta-4d\delta^3+6ga\delta+2h\delta$.

На рис. 4.8 представлено крива щільності розподілу $\varphi(x)$ ймовірності часу виконання технології тестування вразливості до *SQL*-ін'єкцій для приведених вище умов (як вхідні дані використовувалися корені полінома (4.17)).

Зовнішній вигляд графіку дає підстави припустити, що випадкова величина часу виконання технології тестування вразливості до *SQL*-ін'єкцій відповідає гамма-розподілу (близьке до експоненційного).

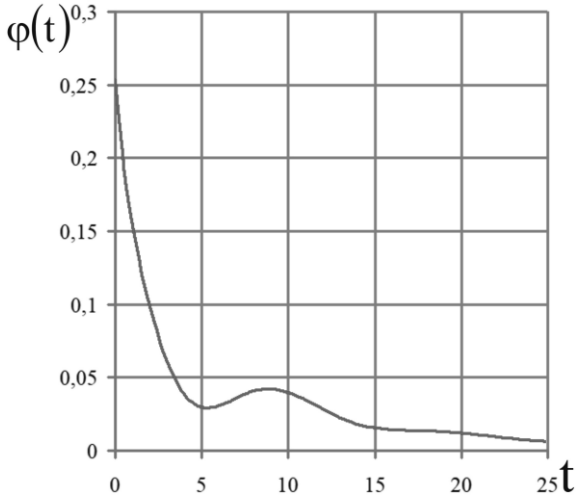


Рис. 4.8. Графік щільності розподілу $\varphi(x)$ ймовірності часу виконання технології тестування вразливості до *SQL*-ін'єкцій

Результати перевірки цієї гіпотези за критерієм $\chi^2$ Пірсона [33] підтвердили її правдоподібність. Так, при досить великому значенні довірчої ймовірності $Q = 0{,}95$ для всіх розглянутих *x1*, *x2* і *x5* відповідні значення дорівнюють $\chi^2$ ($\chi_1^2 = 19{,}3$, $\chi_2^2 = 15{,}1$, $\chi_5^2 = 25{,}6$) $<< \overline{\chi^2} = 101{,}9$.

## 4.3 Висновки до розділу

У розділі розроблено комплекс математичних моделей процесу тестування *Web*-застосунків. В основу математичного моделювання покладено підхід мережевого *GERT* моделювання. В результаті розроблено математичні моделі технології тестування *DOM XSS* вразливості і технології тестування вразливості до *SQL*-ін'єкцій.

Математична модель технології тестування комплексу *DOM XSS* вразливостей відрізняється від відомих урахуванням специфіки комплексного аналізу різних типів *XSS* вразливості ("*stored XSS*", "*reflected XSS*" і *DOM Based XSS*), а також включенням в алгоритм процедур

автоматичного аудиту *DOM Based XSS* окремо. Це дає можливість провести аналітичну оцінку часових витрат тестування вказаних вразливостей в умовах реалізації стратегії розроблення безпечного програмного забезпечення.

Математична модель технології тестування вразливості до *SQL*-ін'єкцій відрізняється від відомих вдосконаленим способом визначення відстані між результатами ін'єкції. Використання в запропонованому способі критерію Джаро-Вінклера для порівняння результатів ін'єкції *SQL*-коду і введення порогового значення дозволить підвищити точність результатів тестування безпеки програмного забезпечення.

В ході дослідження представленого комплексу математичних моделей було визначено, що випадкова величина часу виконання даного процесу тестування в цілому відповідає гамма-розподілу. Перевірка цієї гіпотези проведена за критерієм $\chi^2$ Пірсона.

Отпубліковані наукові результати роботи автора приведені у 4 розділі [9, 10, 13, 14, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52].

**Список використаних джерел:**

[1] Липаев В.В. Надежность и функциональная безопасность комплексов программ реального времени. Москва, 2013. 176 с.

[2] НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп’ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999р., №22. [Електронний ресурс]. – Режим доступу до ресурсу: http://do.gendocs.ru/docs/index-27508.html?page=8

[3] Шибанов, А.П. Обобщенные GERT-сети для моделирования

протоколов, алгоритмов и программ телекоммуникационных систем: дис. доктора техн. наук: 05.13.13 [Текст] / Шибанов Александр Петрович. – Рязань, 2003. – 307 с.

[4] Denning P.J. Discussing Cyber Attack / P.J. Denning, D.E. Denning // Comm. of the ACM. – 2010. – Vol. 53. – №. 9. – P. 29-31.

[5] Столлингс В. Криптография и защита сетей: принципы и практика / В.Столлингс. – М.:Вильямс, 2001.– 672 с.

[6] Han. J. Data mining, southeast Asia edition: Concepts and techniques / J. Han, M. Kamber, J. Pei – Morgan Kaufmann, 2006.– 703 p.

[7] OSSTMM 3. Open-Source Security Testing Methodology Manual [Електронний ресурс] – Режим доступу: http://www.isecom.org/mirror/OSSTMM.3.pdf

[8] Patcha, A. An overview of anomaly detection techniques: Existing solutions and latest technological trends / A. Patcha, J. M. Park : Computer Networks. – 2007. – Vol. 51, №. 12. – pp. 3448-3470.

[9] Коваленко А.В. Алгоритм анализа DOM XSS уязвимости при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез дев'ятнадцятого міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кропивницький 7-8 квітня 2017 р. – Кропивницький: ГЛА НАУ. – 2017. – С. 125-127.

[10] Коваленко А.В. Тестирование уязвимости Web-приложений к атаке вида межсайтовый скриптинг / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов, // Збірник тез «Securitea informationala 2018». Conferenta internationala (editia a XIV-a). Chisinau. Moldova. 20-21 martie 2018. – Chisinau: ADSEM. – 2018. – P. 54-56.

[11] Kovalenko Oleksandr, The mathematical model of the testing technology

for Dom Xss vulnerabilities / O. Kovalenko, O. Smirnov, A. Kovalenko, S. Smirnov, V. Vialkova // Scientific & practical cyber security journal (SPCSJ) Volume 2 Issue 1, P. 22-28. Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2018 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/04-3-o.kovalenko-a.kovalenko-o.smirnov-s.smirnov-v.vialkova.pdf

[12] Testing for DOM-based Cross-site scripting (OTG-CLIENT-001) – OWASP: [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001)

[13] Kovalenko O.V. Method of testing the DOM XSS vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii // International Conference «information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. – 2017. – P. 7.

[14] Коваленко О.В. Метод тестування DOM XSS уразливості / О.В. Коваленко, О.А. Смірнов, А.С. Коваленко, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

[15] Требования к средствам защиты конфиденциальной информации [Електронний ресурс]. – Режим доступу до ресурсу: http://www.sec4all.net/statea118.html

[16] Fowler M. Inversion of Control Containers and the Dependency Injection pattern: [Електронний ресурс] – Режим доступу: https://martinfowler.com/articles/injection.html

[17] Intrusion detection and firewall security. Oslo, 2016. [Електронний ресурс]. – Режим доступу: https://www.cs.hioa.no/teaching/ materials/MS004A/html/L65.en.pdf

[18] Lee.T. Microsoft Corp. Behavioral Classification /Tony Lee & Jigar J. Mody // Presented on the EICAR Conference – 2006. − Режим доступу: http://www.microsoft.com/downloads/details.aspx?FamilyID=7b5d8cc8-b336-4091-abb5-2cc500a6c41a&DisplayLang=en.

[19] The State of the Art in Intrusion Prevention and Detection [Electronic resourse]/ Al-Sakib Khan Pathan. New York : Auerbach Publications, 2014. 516 p. [Електронний ресурс]. – Режим доступу: http://docshare03 .docshare.tips/files/20579/205795770.pdf.

[20] Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник НТУ «ХПІ». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

[21] Zhang Peng. Security in Network Coding / Peng Zhang, Chuang Lin. – Springer, 2016. – 98 p.

[22] Lin W-C. An intrusion detection system based on combining cluster centers and nearest neighbors/ W-C. Lin, S-W. Ke, C-F. Tsai : Knowledge-Based Systems.– 2015. – vol. 78.– pp. 13-21.

[23] Seacord Robert Secure Coding in C and C++ / Robert C. Seacord – Addison-Wesley, 2013. – 600 p.

[24] Seacord Robert The CERT C Coding Standard, Second Edition: 98 Rules for Developing Safe, Reliable, and Secure / Robert C. Seacord – Addison-

Wesley, 2014. - pp 576.

[25] Халіфе К. Gert-модель прогнозування параметрів функціональної безпеки технічних систем / Кассем Халіфе, С.Г. Семенов, С.Ю. Гавриленко // Зб. наукових праць. Системи обробки інформації. – Х.: ХУ ПС, 2016. – Випуск 2(139). – С.50-52.

[26] Халифе К. GERT-модель процесса безопасного тестирования программного обеспечения / Кассем Халіфе, А.Е. Горюшкіна, В.М. Зміївська // Зб. наукових праць. Системи обробки інформації. – Х.: ХУ ПС, 2016. – Випуск 3(140). – С.21-24.

[27] Khalife Kassem Development of Gert model of management system by using test cases / Kassem Khalife, S. G. Semenov, V N. Zmiyevskaya // Journal of Qafqaz university-mathematics and computer science. – 2016. – Vol.(4). – №1 P. 52-59

[28] Pritsker, A. B. GERT: Graphical Evaluation and ReviewTechnique. Part I. Fundamentals / A. B. Pritsker, W. W. Happ The Journal of Industrial Engineering. – 1966. – №5. – P. 681-689.

[29] Третий тип XSS: Межсайтовый скриптинг через DOM [Електронний ресурс]. – Режим доступу до ресурсу: https://www.securitylab.ru/ analytics/275087.php

[30] Смірнов С.А. Метод антивірусного захисту даних з використанням хмарних обчислювальних технологій дис. кандидата техн. наук: 05.13.21 [Текст] / Смірнов Сергій Анатолійович. – Київ, 2017. – 128 с.

[31] Семенов С.Г., Лисица Д.А. Модель оценки риска разработки программного обеспечения: сб. материалов XV Международной НТК «Проблемы информатики и моделирования». Харьков: НТУ «ХПИ», 2015. С.82.

[32] Черушева Т. В. Проектирование программного обеспечения /

Т. В. Черушева. – Пенза : Изд-во ПГУ, 2014. – 172 с.

[33] Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

[34] Testing for SQL Injection (OTG-INPVAL-005) – OWASP: [Електронний ресурс]. – Режим доступу: https://www.owasp.org/ index.php/103 Testing_for_SQL_Injection_(OTG-INPVAL-005)

[35] About The Open Web Application Security Project – OWASP: [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/About_The_Open_Web_Application_ Security_Project

[36] OWASP Top 10 – 2017 RC1: [Електронний ресурс]. – Режим доступу: https://github.com/OWASP/Top10/ blob/master/2017/OWASP%20Top% 2010%20-%202017%20RC1- English.pdf.

[37] Kevin Drebler a , Axel-Cyrille Ngonga Ngomo On the Efficient Execution of Bounded Jaro-Winkler Distances / Semantic Web – Interoperability, Usability, Applicability an IOS Press Journal [Електронний ресурс] – Режим доступу: http://www.semantic-web-journal.net/system/files/ swj944.pdf

[38] Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

[39] Коваленко А.В. Задачи распознавания ситуаций в ERP системах/ А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

[40] Коваленко О.В. GERT-мережевий синтез технології тестування на вразливість WEB-додатків/ О.В. Коваленко // Захист інформації. – Випуск 20(2) – К.: НАУ. – 2018. – С. 89-94.

[41] Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141.

[42] Коваленко О.В. Математичні моделі технології тестування DOM XSS вразливості та вразливості до SQL ін'єкцій / О.В. Коваленко // Вісник Черкаського державного технологічного університету. Серія : Технічні науки №4, 2018. – С. 29-36.

[43] Коваленко О.В. Математична модель технології тестування вразливості до SQL-ін’єкцій / О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 6 (58). – Полтава: ПолтНТУ. – 2019. – С. 43-47.

[44] Коваленко А.В. Технология тестирования DOM XSS уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific & practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

[45] Коваленко А.В. Алгоритм анализа уязвимости SQL Injection для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез другої міжнародної науково-технічної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному

світі» (ПНПЗК-2017). м. Харків.10-12 квітня 2017 р. – Харків: НТУ «ХПІ». – 2017. – С. 27.

[46] Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез IX міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. – 2017. – С. 61.

[47] Коваленко О.В. GERT-модель технології тестування DOM XSS уразливості / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць IV міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 21-24 лютого 2018 р. – Київ: Європейський університет. – 2018. – С. 65-70.

[48] Коваленко О.В. Технології тестування уразливостей Web-застосунків з використанням GERT-моделі / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". м. Кривий Ріг. 21-23 березня 2018 р. – Кривий Ріг.: ДВНЗ КНУ – 2018. – С. 227-230.

[49] Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов, // Збірник тез X міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

[50] Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL-ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

[51] Коваленко А.В. Технология тестирования уязвимости к SQL инъекциям / А.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 5 (45). – Полтава: ПолтНТУ. – 2017. – С. 66-71.

[52] Коваленко О.В. Математична модель технології тестування комплексу DOM XSS вразливостей для аналітичної оцінки часових витрат / О.В. Коваленко // Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 173-180, 2019.

# РОЗДІЛ 5
# РОЗРОБЛЕННЯ ІМІТАЦІЙНОЇ МОДЕЛІ ТЕХНОЛОГІЇ ТЕСТУВАННЯ БЕЗПЕКИ

У розділі розроблено імітаційну модель технології тестування безпеки на основі положень теорії масштабування імітаційних моделей, що відрізняється від відомих адаптацією вибору вхідних операторів управління і даних до підвищення вимог оперативності розроблення і реалізації моделі, яке виразилося в реалізації процедури взаємодії з реальним браузером з використанням засобів автоматизації браузеру і формуванні даних для атаки на декількох діалектах.

## 5.1 Масштабування імітаційної моделі технології тестування безпеки

### 5.1.1 Постановка завдання масштабування імітаційної моделі технології тестування безпеки

Відомим фактом, підтвердженим великою кількістю актуальних результатів досліджень, є доцільність і актуальність проведення імітаційного тестування з використанням комп'ютерних і телекомунікаційних засобів. Для оцінки результатів математичного моделювання технологій тестування безпеки *Web*-застосунків розробимо і вдосконалимо імітаційну модель.

Проведені дослідження показали, що одним з характерних факторів, що впливають на ефективність розроблюваної імітаційної моделі, є істотна залежність від часу реалізації і експерименту. В той же час, більшість існуючих імітаційних моделей мають ряд недоліків, пов'язаних із зайвими витратами обчислювальних ресурсів і часу.

У разі моделювання в реальному часі це призводить до зниження точності результатів. Тому виникає необхідність масштабування імітаційної моделі, яке дозволило б знизити обчислювальну, алгоритмічну, технологічну або інші види складності її аналізу без втрати точності моделювання поведінки на заданому рівні абстракції.

Аналіз літератури показав, що нині існує декілька типових ситуацій, коли в процесі імітаційного моделювання можливе використання операцій масштабування. Наприклад, при перевірці властивостей, описаних на більш високому рівні абстракції, ніж сама модель, або при перевірці локалізованих властивостей (наприклад, властивостей одного з компонентів великої моделі).

Проведені дослідження показали, що більшість авторів [1-8] у першому випадку, як правило, модель перебудовують вручну на необхідному (більш високому) рівні абстракції. У другому випадку детальні моделі компонентів, перевірка властивостей яких не передбачається або вважається надмірною, замінюються на "нульові компоненти".

Надалі для валідації таких спрощених компонентів використовуються знання експертів. Однією з основних проблем такого підходу є оцінка міри адекватності і можливості таких спрощень.

В той же час, як показали дослідження, динамічний вибір достатнього рівня моделювання (міри масштабування) безпосередньо в ході експерименту може усунути цей недолік.

Для аргументованого вибору і розроблення способу масштабування проведемо аналіз існуючих підходів і алгоритмів.

Проведені дослідження показали, що нині існують різні види залежності між операторами імітаційної моделі. Це транзитивні залежності за даними й управлінням.

Ці види залежності описані в літературі [9-11]. У дисертаційній роботі пропонується використати канонічне визначення залежності за даними, що зв'язує два оператори послідовного процесу з уточненням відмінності за значенням змінної.

Так само для врахування залежності за управлінням пропонується використати результати роботи [11] з уточненням, що враховує залежності, які виникають при зацикленні ділянки програми.

Дещо відособлено у списку залежностей стоять залежності за часом виконання, якщо модельний час виконання одного з них залежить від модельного часу виконання іншого.

При цьому треба враховувати, що не всі оператори імітаційної моделі просувають модельний час [8]. У дисертаційній роботі цим видом залежності було вирішено нехтувати, у зв'язку з відсутністю технологічної необхідності.

### 5.1.2 Алгоритми масштабування імітаційної моделі технології тестування вразливостей

Основні визначення.

Визначення 5.1. Вершина $n$ є батьком деякої вершини $m$ (нащадка) в графі $G = (N, E, n_0)$, якщо $(n, m) \in G.E$.

Множину усіх нащадків вершини $n$ в графі $G$ будемо позначати як $desc(n,G)$.

Визначення 5.2. Шляхом «*way*» з $n_i \in G.N$ до $n_k \in G.N$ називається послідовність вершин $n_i$, $n_{i+1}$ ,..., $n_k$ така, що будь-які дві сусідні вершини в ній пов'язані дугою в графі:

$$(n_j, n_{j+1}) \in G.E,\ j = i,\ k-1.$$

Запис $n \in$ «*way*» означає, що вершина $n$ зустрічається у шляху «*way*».

Визначення 5.3. Шлях «*way*» називається простим, якщо він складається з однієї вершини.

Визначення 5.4. Максимальним називається шлях, який нескінченний, або закінчується у вершині, що не має нащадків.

Масштабування за управлінням.

У основі масштабування за управлінням лежать постулати, що характеризують чутливість операторів до нескінченного зациклення через поняття максимального шляху [12], в термінах послідовних процесів логічної схеми імітаційної моделі.

Визначення 5.5. У графі управління $G$ послідовного процесу $p$ оператор $n_j \in G.N$ прямо залежить за управлінням чутливо до зациклення від оператора $n_i \in G.N$ тоді і лише тоді, коли у $n$ є два нащадки $n_k$ і $n_z$ такі, що:

1) в усіх максимальних шляхах, що починаються з $n_k$, зустрічається $n_j$, і або $n_i = n_j$, або $n_j$ строго передує будь-якому входженню $n_i$;

2) існує максимальний шлях, що починається з $n_z$, такий, що або в ньому не зустрічається $n_j$, або $n_i$ строго передує будь-якому входженню $n_j$.

У роботі [13] наводиться узагальнений алгоритм побудови графа залежностей за управлінням. Проте, проведені дослідження показали, що для даного завдання імітаційного моделювання технології пошуку вразливостей не потрібно знаходження прямої залежності за управлінням.

Для коректного моделювання даного процесу досить використати слабкіше поняття транзитивної залежності за управлінням. Це істотно знизить обчислювальну складність алгоритму масштабування.

На рис. 5.1. представлено блок-схему алгоритму обчислення транзитивної залежності за управлінням. Слід зауважити, що в роботі [7, 8] доведено коректність застосування транзитивної залежності за управлінням для масштабування.

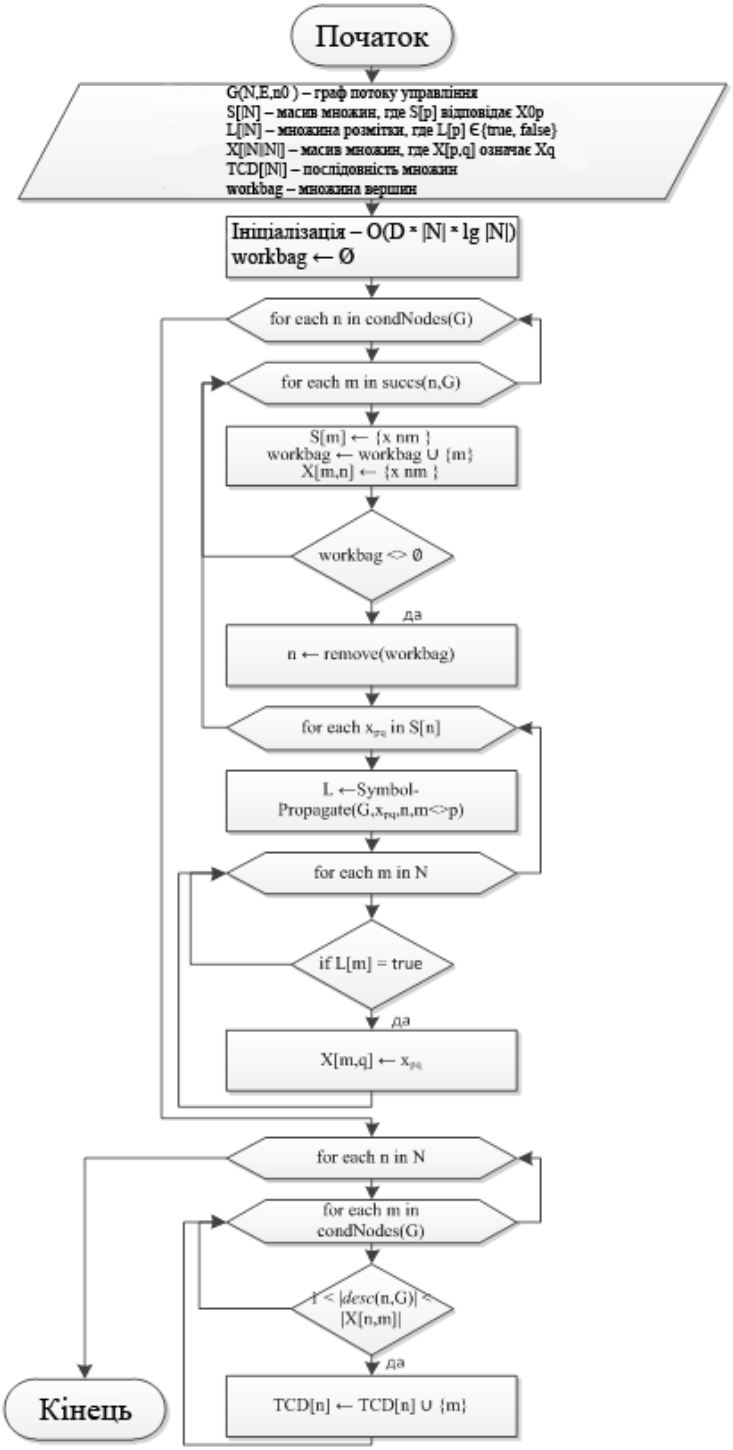


Рис. 5.1. Блок-схема алгоритму обчислення транзитивної залежності за управлінням

Проведені розрахунки показали, що загальна складність алгоритму дорівнює O(D 2 × |N| 2 ). Порівняльна оцінка запропонованого алгоритму з відомим алгоритмом, описаним в [12], показала зменшення складності за

рахунок заміни циклу по всіх керуючих вершинах з множиною символів розміру D×|N|.

Масштабування за даними.

Як вже було вказано вище, залежність за даними описується в роботах [14-16]. У дисертаційній роботі використаємо формалізацію відомих визначень, уточнимо її вказуванням змінної, за значенням якої виникає залежність.

Визначення 5.6 В графі управління $G$ послідовного процесу $p$ оператор $n_j \in G.N$ залежить за даними від оператора $n_i \in G.N$ по змінній $v$ тоді, коли існує змінна $v \in p.V_{ar}$ така, що:

1) існує непростий шлях «*way*» з $n_i$ до $n_j$ такий, що для будь-якого $n_k \in$ «*way*» $-\{ n_i, n_j \}$,

2) $v \in p.def(n_i) \cap p.ref(n_j)$.

На рис. 5.2. представлена блок-схема алгоритму обчислення транзитивної залежності за даними. Оцінка і розрахунки обчислювальної складності представленого алгоритму показали, що загальна складність алгоритму складе O(|V | × |N| 2 ).

Скористаємося даними алгоритмами для вдосконалення імітаційної моделі технології тестування вразливостей.

## 5.2 Імітаційна модель технології тестування безпеки *Web*-застосунків

### 5.2.1 Загальні вимоги і структура імітаційної моделі

Запропонований в роботі підхід до масштабування імітаційної моделі ґрунтується на положеннях відомих теорій масштабування [11], у яких зафіксовані основні етапи.

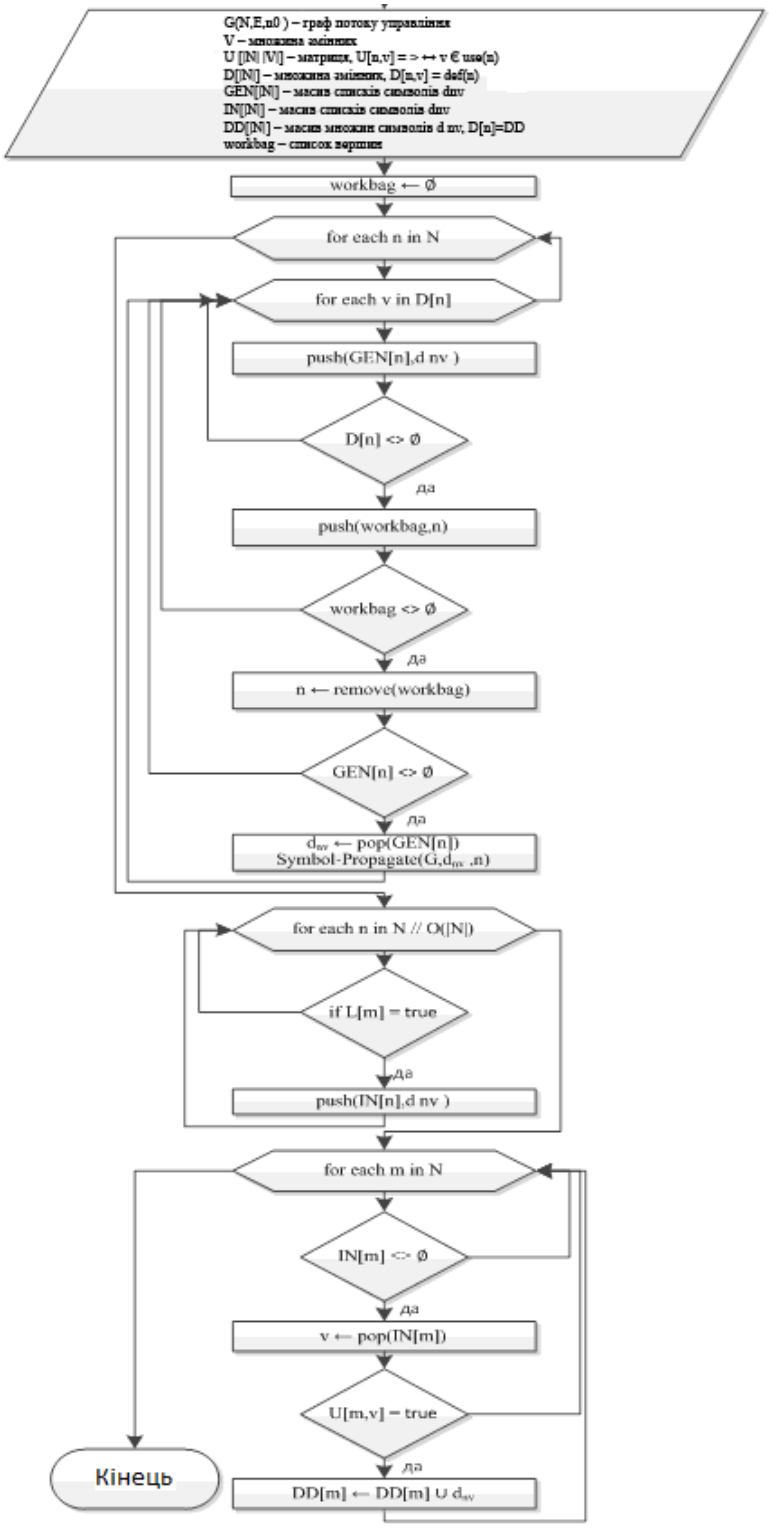


Рис. 5.2. Блок-схема алгоритму обчислення транзитивної залежності за даними

Зокрема, це виявлення операторів, що не впливають на генерацію і хід подій, спостереження яких потрібно для перевірки заданих властивостей

поведінки системи (незначимих операторів) і абстрагування опису імітаційної моделі від таких операторів [12].

В розроблюваній імітаційній моделі технології тестування вразливостей пропонується не лише просте абстрагування від несуттєвого оператора, а його заміна на ефективніший, з точки зору точності кінцевих результатів перевірки, оператор.

Для імітаційної моделі технології тестування вразливостей *Web*-застосунків розроблено програмний застосунок, що проводить атаки, на які вказуватиме надане користувачем *URL*-посилання по наведених алгоритмах, і що аналізує результат атак на наявність певної вразливості у *Web*-застосунку.

Як було вказано вище, для зниження часових витрат на імітаційне моделювання було вирішено провести масштабування шляхом заміни процедур інформаційного обміну з сервером за допомогою *HTTP*-клієнта на процедури взаємодії з реальним браузером з використанням засобів автоматизації браузеру.

Окрім основної мети масштабування ця заміна дозволить підвищити достовірність результату тестування атаки з ін'єкцією *Javascript* коду. При цьому як засіб автоматизації браузеру було обрано бібліотеку *Selenium Webdriver*.

Аналіз літератури показав, що *Selenium Webdriver* є інструментом автоматизації браузеру, який дозволяє розробляти програмні продукти, що мають можливість управляти поведінкою браузеру [17-19]. Для роботи з *Selenium Webdriver* потрібні наступні програмні компоненти:

– *Web*-браузер, який буде автоматизований;

– драйвер для встановленого браузеру, який дозволяє автоматизувати поведінку браузеру;

– безпосередньо програмний продукт, який складається з набору команд певної мови програмування для драйвера браузеру, що надаються спеціальними бібліотеками для багатьох мов програмування.

При реалізації імітаційної моделі в якості браузеру було обрано *Web*-браузер *Google Chrome*. *Google Chrome* - браузер, розроблений компанією *Google* на основі браузеру з відкритим кодом *Chromium* та іншого відкритого програмного забезпечення.

Цей вибір обумовлений тим, що компанією *Google* в стратегії розвитку архітектури визначений факт, що сьогодні більшість веб-сайтів є не просто веб-сторінками, а веб-застосунками.

В якості драйвера браузеру використаний *ChromeDriver*, який є імплементацією *Webdriver* для браузеру *Google Chrome*.

Для розроблення програмного продукту, який управлятиме браузером і виконуватиме атаки на *Web*-застосунки, необхідно обрати мову програмування.

Для реалізації необхідного функціонала до мови програмування висунено наступні вимоги:

– наявність бібліотеки команд *Selenium Webdriver* для цієї мови програмування;

– можливість роботи з *HTTP* за допомогою нативних або сторонніх *HTTP* клієнтів;

– наявність нативної або сторонньої системи журналювання з виведенням логу в консоль та зі збереженням логу в текстовому файлі.

Також до мови програмування висунено наступні вимоги для підвищення якості результуючого продукту і підвищення ефективності процесу програмування:

– незалежність мови програмування від оточення, де виконуватиметься програмний продукт;

– наявність сторонніх бібліотек загального призначення;

– наявність інструменту для автоматизованого збору програми і управління зовнішніми залежностями.

Існує декілька мов програмування, що задовольняють цим вимогам, наприклад, *Python, Ruby, Java*. Проте, для програмної реалізації методу тестування вразливостей *Web*-застосунку обрана мова програмування *Java*.

Аналіз літератури показав, що мова програмування *Java* відповідає вимогам до незалежності від оточення за допомогою компіляції початкового *Java*-коду у байткод, який є спрощеними машинними командами.

Потім програму можна виконати на будь-якій платформі, що має встановлену віртуальну машину *Java*, яка інтерпретує байткод в код, пристосований до специфіки конкретної операційної системи і процесора [17]. Зараз віртуальні машини *Java* існують для більшості процесорів і операційних систем.

Для збору програм, розроблених з використанням мови програмування *Java*, існує три найбільш поширених інструменти автоматизованого збору - *Apache Ant, Apache Maven* і *Gradle*. Для реалізації цього програмного продукту обрано інструмент *Apache Maven*.

*Apache Maven* - це засіб автоматизації роботи з програмними проектами, який використовується для *Java*-проектів для управління і збору програм. Для опису програмного проекту, який треба побудувати (*build*), *Maven* використовує конструкцію, відому як *Project Object Model (POM)*, залежності від зовнішніх модулів, компонентів і порядку побудови. *Maven* базується на плагін-архітектурі, дозволяє зробити використання будь-якої програми контрольованим через стандартний вхід.

Цей інструмент має не лише функціонал зі збору програм, а і з управління зовнішніми залежностями і розбиття програмного продукту на окремі незалежні модулі з подальшим збором в єдиний модуль [20, 21].

Стандартна бібліотека *Java* надає можливість роботи з мережами, зокрема, протоколом *HTTP*. Також стандартна бібліотека *Java* містить класи для роботи текстом, *URL*, колекціями даних та іншими функціями загального призначення.

Для мови програмування *Java* існує бібліотека команд *Selenium Webdriver*. Найбільш актуальну версію можна отримати з репозиторія *Maven* як окрему бібліотеку або використати її як зовнішню залежність в *Java* проекті.

Мова програмування *Java* має велику кількість бібліотек для журналу з можливостями розширеної конфігурації. Для використання в програмній реалізації методу тестування вразливостей *Web*-застосунків обрано бібліотеку журналу *Apache Log4j2*.

Бібліотека *Apache Log4j2* має велику кількість функцій і можливостей конфігурації журналу, а саме можливість переадресації журналів виконання програми до будь-якого потоку виведення, наприклад, в текстовий файл на екран, в мережевий сокет та ін. [21].

Таким чином, мова програмування *Java* задовольняє усім вимогам, які були висунені до мов програмування для можливості реалізації методу тестування вразливостей *Web*-застосунків.

Слід зазначити, що для розроблення застосунку буде використано останню версію *Java*, а саме *Java* 8.

Для розроблення структури програмної реалізації методу тестування вразливостей *Web*-застосунків необхідно позначити великою кількістю функціональних вимог, які повинен реалізовувати застосунок:

– отримання вхідних параметрів, а саме типу вразливості для тестування і *URL* веб-сторінки для тестування в якості параметрів командного рядка;

– для типу вразливості *DOM XSS*, програмний продукт повинен отримати код окремої веб-сторінки і код усіх зовнішніх *Javascript*-файлів, які використовуються. Після цього провести статичний аналіз усього присутнього *Javascript*-коду на наявність певних маркерів, які можуть призводити до вразливості;

– для кожного *GET*-параметру в *URL*-сторінці треба визначити тип рефлексії параметра на сторінці і провести *XSS* атаку з тими, що відповідають для типу рефлексії даними, і перевірити її результат;

– для типу вразливості *SQL Injection*, програмний продукт повинен для кожного *GET*-параметру в *URL*-сторінці провести *Boolean blind based SQL*-ін'єкцію і визначити її результат на основі аналізу тексту веб-сторінок, які повертає сервер;

– детальне логування ходу роботи програмного продукту на консоль та у файл.

Враховуючи ці вимоги до застосунку, загальна структура програмної реалізації виглядатиме як багаторівневий проект з наступними модулями:

– модуль інтерфейсу користувача;

– модуль аналізу вразливостей *DOM XSS*;

– модуль аналізу вразливості до *SQL*-ін'єкції;

– модуль функцій загального призначення.

### 5.2.2 Структура *Maven*-проекту

З основних джерел літератури [20-21], використовуваних в дисертаційній роботі відомо, що *Java*-программа є набором скомпільованих *Java*-класів, зібраних у виконуваний архів типу *.jar*.

Проведений аналіз літератури показав, що для збирання програмного продукту в *.jar* файл існує декілька способів. У цьому проекті в якості інструменту збору проекту був вибраний *Apache Maven*.

Дослідження показали, що життєвий цикл збирання *Maven*-проекту містить фіксований набір фаз, що виконуються одна за однією. Життєвий цикл за змовчуванням складається з наступних фаз:

– *validate* (виконується перевірка коректності проекту з точки зору *Maven*);

– *compile* (виконується компіляція файлів початкового коду);

– *test* (здійснюється тестування скомпільованого коду за допомогою наданого набору модульних тестів);

– *package* (реалізується упаковування скомпільованого коду в певну форму дистрибуції, наприклад, в *JAR*-архів);

– *install* (переміщується результат упаковування в локальний репозиторій для використання в локальних проектах);

– *deploy* (переміщається результат упаковування до віддаленого репозиторію для використання іншими розробниками).

*Apache Maven* використовує один або декілька файлів *pom.xml* для представлення у форматі *XML*-структури *Java*-проекту. *Project Object Model (POM)* містить інформацію про структуру проекту, його внутрішні і зовнішні залежності, інформацію про процес збирання проекту і плагінів, які будуть використані при певних фазах збирання проекту [20].

*Maven* підтримує структуру багатомодульних проектів за допомогою наслідування і агрегації проектів. Це досягається введенням додаткових артефактів із спеціальним маркером у файлах *pom.xml*.

При використанні такої конфігурації, загальні характеристики і налаштування властивостей можна виносити у базовий файл *pom.xml*, який буде успадкований проектами-нащадками.

Мінімальними необхідними елементами файлу *pom.xml* є координати проекту, містять наступні елементи:

– *groupId* - унікальний ідентифікатор проекту або організації, яка розробляє проект. Використовується як аналог пакету в мові *Java*.

– *artifactId* - унікальний ідентифікатор артефакту в проекті. Артефактом може бути як сам проект, так і модуль багатомодульних проектів.

– *version* - версія артефакту. Використовується для версіонування артефактів в репозиторії.

– *packaging* - тип артефакту проекту *Maven*. У загальному випадку має значення *jar* для звичайних проектів або окремих модулів, і *pom* для багатомодульних проекту (агрегатора).

Для програмної реалізації методу тестування вразливостей *Web*-застосунків було обрано багатомодульну структуру, отже, *Maven*-проект також буде розроблений як багатомодульний проект з декількома модулями. Таким чином, враховуючи загальну структуру застосунку, *Maven*-проект складатиметься з кореневого проекту, проекту модулю інтерфейсу, проекту модулю функцій загального призначення, кореневого проекту модулів аналізу вразливостей і безпосередньо проектів модулів аналізу вразливостей. Запропонована структура *Maven*-проекту є розширюваною, оскільки наявність кореневих проектів дозволяє легко під'єднати додаткові модулі аналізу вразливостей.

Кореневий *Maven*-проект (*dxss-sqli-framework*) визначає ідентифікатор групи для всіх дочірніх проектів як *com.kntu.mtf.koval*. Проект є агрегатором, тому параметр *packaging* має значення *pom*.

```
<groupId> com.kntu.mtf.koval. </groupId>
<artifactId>dxss-sqli-framework</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>pom</packaging>
```

Кореневий проект є агрегатором модулів, тобто проект містить посилання на всі дочірні проекти, використовувані в застосунку.

```
<modules>
    <module>framework-core</module>
    <module>attack-modules</module>
    <module>core-utils</module>
</modules>
```

Проект також містить залежності, використовувані всіма дочірніми проектами, а також вказівку *Maven* використати версію 8 мови програмування *Java* при компіляції початкових файлів.

```
<build>
    <plugins>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.1</version>
        <configuration>
            <source>1.8</source>
            <target>1.8</target>
        </configuration>
    </plugin>
    </plugins>
</build>
```

Проект модуля функцій загального призначення називається core-utils. Проект має тип упаковування *jar* і являється дочірнім до проекту *dxss-sqli-framework*.

```
<artifactId>core-utils</artifactId>
<packaging>jar</packaging>
<parent>
    <artifactId>dxss-sqli-framework</artifactId>
    <groupId>com.kntu.mtf.koval.</groupId>
```

```
        <version>1.0-SNAPSHOT</version>
    </parent>
```

Кореневий проект модулів аналізу вразливостей називається *attack-modules*. Проект є агрегатором, тому параметр *packaging* має значення *pom*. Цей проект являється дочірнім для *dxss-sqli-framework* і містить посилання на проект *core-utils* як залежність.

```
    <artifactId>attack-modules</artifactId>
    <packaging>pom</packaging>
    <parent>
        <artifactId>dxss-sqli-framework</artifactId>
        <groupId> com.kntu.mtf.koval. </groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modules>
        <module>sqli-module</module>
        <module>dxss-module</module>
    </modules>
    <dependencies>
        <dependency>
            <groupId> com.kntu.mtf.koval. </groupId>
            <artifactId>core-utils</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
```

Проект модуля аналізу вразливості *DOM XSS* називається *dxss-module*. Проект має тип упаковки *jar* і являється дочірнім до проекту *attack-modules*.

Проект модуля аналізу вразливості до *SQL*-ін'єкцій називається *sqli-module*. Проект має тип упаковки *jar* і являється дочірнім до проекту *attack-modules*.

Проект модулю інтерфейсу називається *framework-core*. Проект має тип упаковки *jar*, являється дочірнім для *dxss-sqli-framework* і містить посилання на проекти *sqli-module* і *dxss-module* як залежності.

Загальна ієрархія проектів у багатомодульних *Maven*-проектах представлена на рис. 5.3.

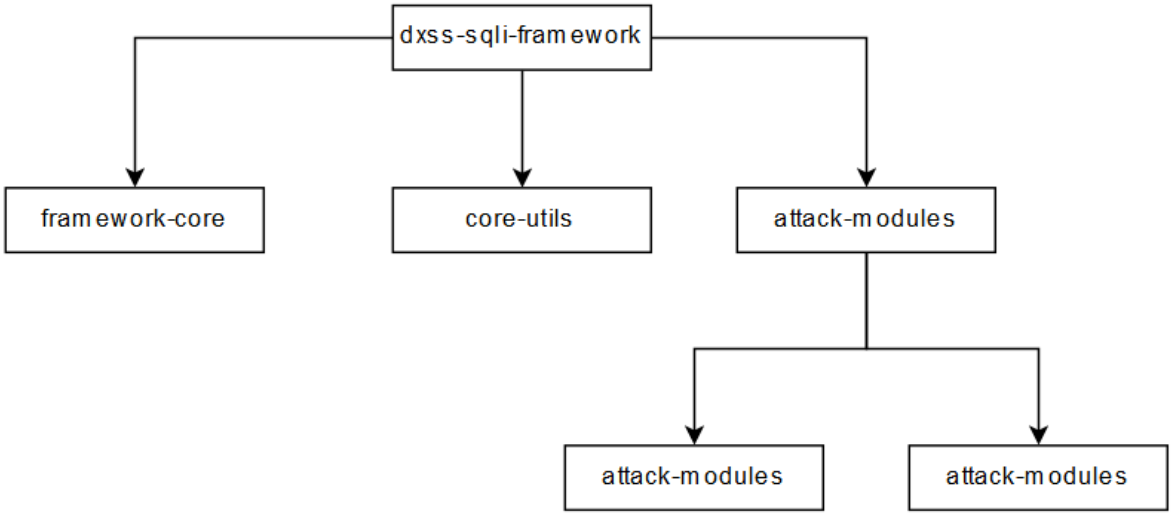


Рис. 5.3. Ієрархія проектів у багатомодульному *Maven*-проекті

### 5.2.3 Структура модуля інтерфейсу

У розробленій імітаційній моделі технології тестування вразливостей модуль інтерфейсу відповідає за запуск застосунку, обробку параметрів командного рядка, і виконання тесту *Web*-застосунку на наявність певної вразливості залежно від переданих параметрів командного рядка.

Для запуску застосунку, розробленого на мові програмування *Java*, потрібен клас, який містить статичний метод *main*, аргументами до якого передаються параметри командного рядка.

У програмній реалізації імітаційної моделі методу тестування вразливостей *Web*-застосунків цей клас називається *Application*. Цей клас

передає параметри командного рядка іншим класам на обробку і обробляє виняткові ситуації під час виконання програми.

Для обробки параметрів командного рядка з метою максимальної масштабованості використаний шаблон проектування "Команда". Команда - це шаблон проектування, який відноситься до класу шаблонів поведінки [23].

Він інкапсулює запит у формі об'єкту, що, у свою чергу, дозволяє використати єдиний інтерфейс виконання різних дій.

Цей шаблон проектування доцільно використовувати при роботі з командним рядком, коли є більш ніж один варіант роботи застосунку. *UML*-діаграма шаблону проектування "Команда" приведена на рис. 5.4.

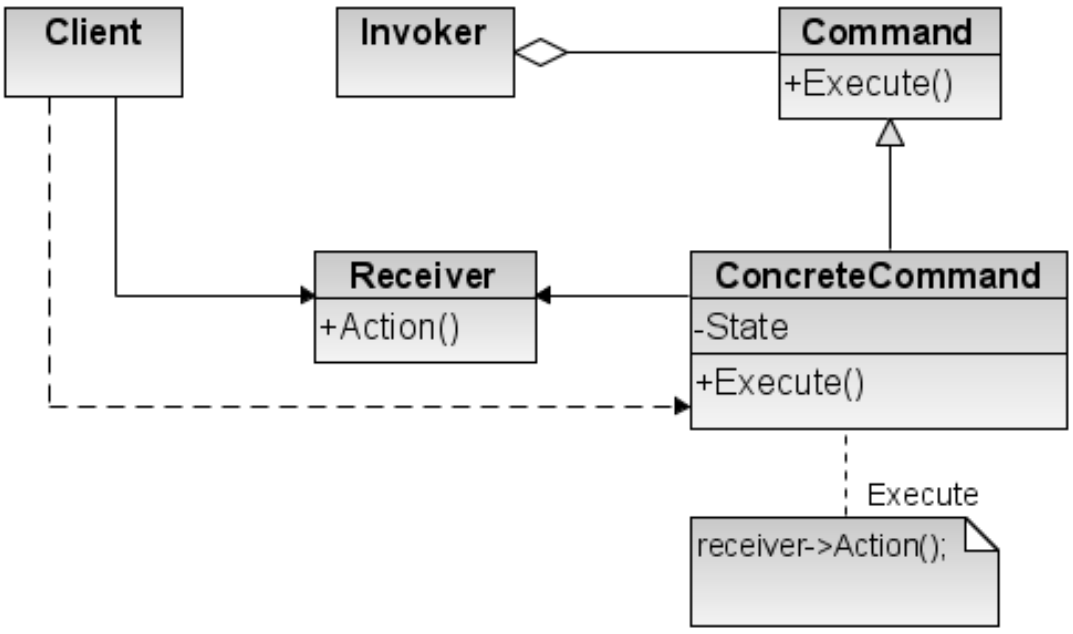


Рис. 5.4. *UML*-діаграма шаблону проектування "Команда"

Структурні елементи шаблону виконують наступні функції:

– *Command* - оголошує інтерфейс виконання операції;

– *ConcreteCommand* - реалізує конкретну команду, викликаючи певні методи об'єкту *Receiver*;

– *Client* - створює об'єкт *ConcreteCommand* і встановлює його одержувачів;

– *Invoker* - звертається до команд з метою виконання.

*UML*-діаграма реалізації цього шаблону проектування приведена на рис. 5.5.

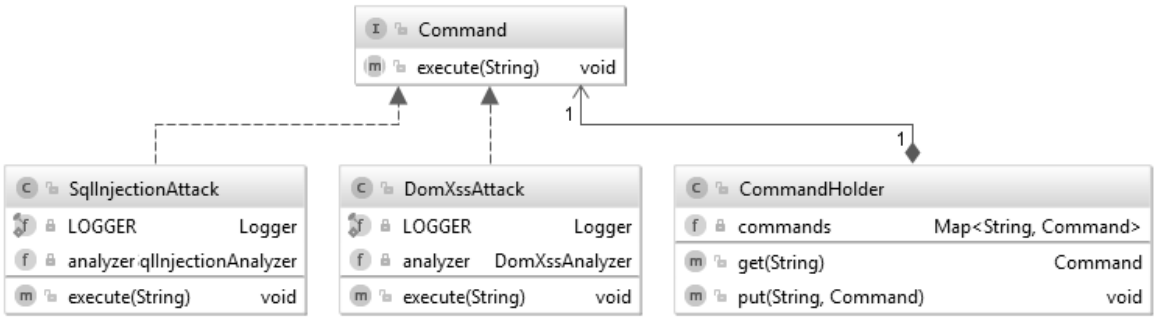


Рис. 5.5. *UML*-діаграма реалізації шаблону проектування "Команда"

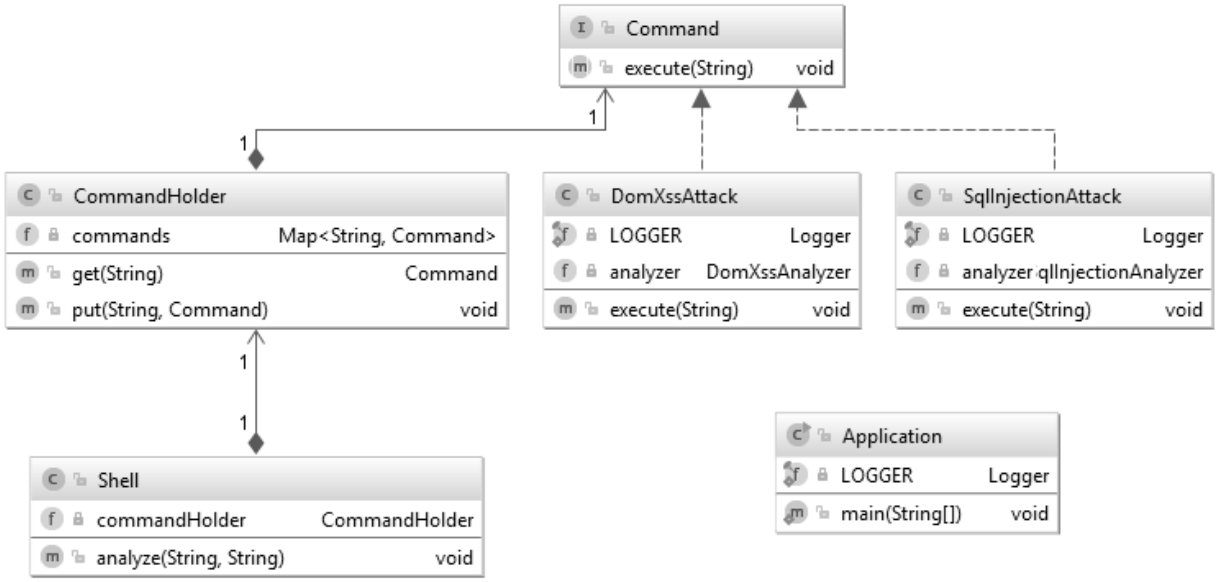


Рис. 5.6. Загальна *UML*-діаграма модуля інтерфейсу

У цій реалізації шаблону "Команда" є присутнім інтерфейс *Command*, його імплементації *SqlInjectionAttack* і *DomXssAttack*, а також контейнер команд *CommandHolder*.

В якості структурного елементу *Invoker* шаблону проектування "Команда" виступає клас *Shell*, який приймає параметри командного рядка від класу *Application* і використовує їх для отримання потрібного об'єкту - команди і його виконання. Таким чином, загальна *UML*-діаграма модуля інтерфейсу зображена на рис. 5.6.

### 5.2.4. Розробка модуля інтерфейсу користувача

Модуль інтерфейсу користувача відповідає за запуск додатка, опрацювання параметрів командної строки, та виконання аналізу Web-додатка на наявність певної вразливості залежно від переданих параметрів командної строки.

Першим етапом розробки модуля інтерфейсу користувача є розробка конфігураційного класу контейнера для впровадження залежностей Spring. Даний клас має назву FrameworkCoreConfig та є відповідальним за створення об’єктів класів CommandHolder, DomXssAttack, SqlInjectionAttack та Shell.

```
@Configuration
public class FrameworkCoreConfig {
    @Bean
    public CommandHolder getCommandHolder() {
           CommandHolder holder = new CommandHolder();
           holder.put("sqli", getSqlInjectionAttack());
           holder.put("dxss", getDomXssAttack());
             return holder;
    }
    @Bean
    public DomXssAttack getDomXssAttack()
    {
           return new DomXssAttack();
    }
```

```
@Bean
public SqlInjectionAttack getSqlInjectionAttack()
{
       return new SqlInjectionAttack();
}
@Bean
public Shell getShell()
{
return new Shell();
}}
```

Для реалізації шаблону проектування «Команда» існує інтерфейс Command, який містить єдиний метод execute, що приймає строку з URL Web-сторінки як параметр.

```
public interface Command
{
void execute(String baseUrl);
}
```

Клас DomXssAttack реалізує інтерфейс Command та відповідальний за проведення аналізу Web-сторінки із вказаним URL на наявність вразливості до DOM XSS та журналювання результату аналізу.

Для виконання аналізу в клас у якості залежності впроваджений об’єкт класу DomXssAnalyzer із модуля аналізу вразливості до DOM XSS.

Клас SqlInjectionAttack реалізує інтерфейс Command та відповідальний за проведення аналізу Web-сторінки із вказаним URL на наявність вразливості до SQL ін’єкції та журналювання результату аналізу.

Для виконання аналізу в клас у якості залежності впроваджений об’єкт класу SqlInjectionAnalyzer із модуля аналізу вразливості до SQL ін’єкції.

Даний модуль містить посилання на класи з інших модулів, тому в даному модулі буде створений контекст додатка за допомогою класу -

агрегатора конфігурацій. Даний клас має назву ApplicationConfig та містить посилання на конфігураційні класи усіх модулів.

```
@Configuration
@Import({DomXssConfig.class, FrameworkCoreConfig.class,
        SqlInjectionConfig.class})
public class ApplicationConfig
{
@Bean
public static PropertySourcesPlaceholderConfigurer
       getConfigurer()
       {
       return new PropertySourcesPlaceholderConfigurer();
       }
}
```

Головним класом додатка є клас Application. Даний клас містить метод main, що є точкою входу для додатка. В даному класі створюється програмний контекст Spring за допомогою класу конфігурації ApplicationConfig, після цього із контексту додатка отримується об’єкт Shell, усі залежності якого створені та впроваджені Spring, та виконується обробка параметрів командної строки та виконання додатка.

```
public static void main(String[] args)
{
ConfigurableApplicationContext applicationContext =
New AnnotationConfigApplicationContext
(ApplicationConfig.class);
applicationContext.registerShutdownHook();
Shell shell = applicationContext.getBean(Shell.class);
Try.run(()->{
String url = args[1];
String method = args[0];
shell.analyze(url, method);
```

```
}).onFailure(exception->Match(exception).of(
Case(instanceOf(IndexOutOfBoundsException.class),e->
run(() -> LOGGER.error("Application accepts two
parameters: attack type and URL"))),
Case($(), e -> run(() -> LOGGER.error(e.getMessage())))
));
}
```

### 5.3.5 Розробка модуля аналізу вразливостей DOM XSS

Розробка даного модуля починається з розробки класу конфігурації контексту додатка Spring. Даний клас має назву DomXssConfig та містить методи, що конструюють об’єкти класів DomXssAnalyzer, DomScanner, JavascriptScanner та PayloadGenerator.

Даний клас також містить метод, який створює об’єкт WebDriver для запуску та керування браузером.

```
@Configuration
public class DomXssConfig
{
    @Bean
    public DomXssAnalyzer getDomXssAnalyzer()
{
    return new DomXssAnalyzer();
    }
    @Bean
public DomScanner getDomScanner(){
    return new DomScanner();
    }
    @Bean
public JavascriptScanner getJavascriptScanner() {
   return new JavascriptScanner();
```

```
        }
        @Bean
    public PayloadGenerator getDomXssPayloadGenerator() {
        return new PayloadGenerator();
        }
        @Bean(destroyMethod = "quit")
        @Lazy
        @Qualifier("DxssWebdriver")
        @Scope(SCOPE_SINGLETON)
    public WebDriver dxssWebdriver() {
        System.setProperty("webdriver.chrome.driver",
    DomXssConfig.class.getResource("/driver/chrome_driver.exe").
getPath()
        );
        return new ChromeDriver();
        }
    }
```

Клас DomScanner використовується для сканування HTML коду Web-сторінки на наявність тегів <script> та отримання їх вмісту, якщо скрипт є вбудованим, або посилання на зовнішній файл, якщо скрипт є зовнішнім.

Для аналізу HTML коду Web-сторінки вирішено використовувати потоковий парсер, оскільки це забезпечить більшу швидкість та більш економічне використання пам’яті у випадку великих динамічних сторінок, ніж при використанні DOM парсера.

Для цього була використана технологія SAX, яка дозволяє розробляти легкі в конфігуруванні потокові парсери XML.

Для обробки HTML за допомогою SAX парсера, потрібно розробити клас з певним набором функцій зворотного виклику на певні події під час потокового аналізу вмісту HTML документа.

Для цього був розроблений клас ResponseHtmlHandler, що наслідуваний від класу DefaultHandler бібліотеки SAX.

В даному класі міститься результат парсингу HTML коду сторінки у виглди списку об’єктів класу Javascript, які містять тип тегу <script> та його тіло або посилання на зовнішній файл.

```
public class DomScanner {
public List<Javascript> findJavascript(String htmlContents)
{
return Try .of(() -> {
  InputStream is = new
  ByteArrayInputStream(htmlContents.getBytes());
  ResponseHtmlHandler handler = new ResponseHtmlHandler();
  SAXParserImpl.newInstance(null).parse(is, handler);
  return handler.getJavascriptList();
  }).getOrElseThrow((Function˂Throwable,
  IllegalStateException>)
IllegalStateException::new); }
}
```

Для аналізу отриманого з HTML коду сторінки Javascript коду використовується клас JavascriptScanner. Клас має два методи для роботи з отриманим Javascript кодом.

Метод fetchExternalJavascript звертається до віддалених файлів Javascript за посиланням із атрибута src тега <script> та отримує вміст віддаленого файлу Javascript.

Для цього використовується клас HttpUtils модуля функцій загального призначення, який отримує вміст віддаленого файлу Javascript за допомогою виконання GET запиту за посиланням із атрибута src.

Метод scanJavascript виконує аналіз вмісту тегів <script> на наявність маркерів вразливості у вигляді стоків та витоків. Список стоків та витоків винесений в окремий текстовий файл, що спрощує конфігурацію.

Аналіз Javascript файлів виконується наступним чином. Спочатку, вміст файлу чи тега ділиться на строки, після цього перевіряється входження хоча б одного стоку чи витоку в строку Javascript коду. Результатом роботи методу є список тегів Javascript із наявними стоками чи витоками.

Головним класом модуля є клас DomXssAnalyzer, який відповідає за безпосередньо аналіз Web-сторінки, а саме визначення типу рефлексії GET параметру URL, а також за проведення атаки.

Для визначення типу рефлексії потрібно замінити значення параметру GET запиту в питомій URL на унікальне значення, після чого провести аналіз HTML коду сторінки на наявність даного унікального значення.

```
String uniqueToken = UUID.randomUUID().toString();
String injectedUrl = UrlUtils.replaceParameterValue(
                     url, parameterName, uniqueToken);
```

Залежно від того, в середині якого елемента знаходиться значення, рефлексію поділяють на:

– рефлексію всередині елементу;

– рефлексію в посиланні;

– рефлексію в атрибуті.

```
private ReflectionType detectReflectionType(
   String reflectedString, String token) {
      Matcher attributeRegex =
      Pattern.compile(String.format("<[^>]*%s[^>]*>",
      token)).matcher(reflectedString);
      Matcher anchorHrefRegex = Pattern.compile(
      String.format("<a [^>]*href=[^>]*%s[^>]*>",
      token)).matcher(reflectedString);
      Matcher htmlTagRegex =
      Pattern.compile(String.format
      ("<[^>]*>[^<]*%s[^<]*</[^>]*>",token)).
      matcher(reflectedString);
```

```
        return Match(reflectedString).of(
        Case(e->anchorHrefRegex.matches(),
        ReflectionType.ANCHOR_HREF_ATTRIBUTE),
        Case(e -> attributeRegex.matches(),
        ReflectionType.ATTRIBUTE),
        Case(e -> htmlTagRegex.matches(),ReflectionType.TAG),
        Case($(),ReflectionType.NOT_REFLECTIVE)
        );
    }
```

Після визначення типу рефлексії виконується підбір даних для атаки залежно від типу рефлексії GET параметру URL.

Для цього реалізований клас PayloadGenerator, який містить набори даних залежно від типу рефлексії параметру. Потім, для кожної строки з даними для атаки виконується DOM XSS атака Web-сторінки за вказаним URL. Значення параметру GET запиту замінюється на значення даних для атаки та виконується перехід по отриманій URL в браузері, після чого виконується перевірка на наявність маркеру в DOM структурі сторінки.

Якщо маркер є, це свідчить про наявність вразливості. В даній реалізації у якості маркеру було обрано діалогове вікно Alert браузера.

```
String urlWithPayload = UrlUtils.replaceParameterValue(
                        url, parameterName, payload);
return Try.of(() -> { driver.get(urlWithPayload);
if (ReflectionType.ANCHOR_HREF_ATTRIBUTE.
    equals(reflectionType)){
driver.findElement(By.xpath(String.
format("//a[contains(@href,
'%s')]", payload))).click();}
(new WebDriverWait(driver,5))
.until(ExpectedConditions.alertIsPresent());
return Boolean.TRUE;}).peek(e -> driver.switchTo().
alert().dismiss())
```

```
.recover(e -> Match(e).of(Case(instanceOf
(TimeoutException.class), Boolean.FALSE))).get();
```

### 5.2.6 Розробка модуля аналізу вразливості до SQL ін’єкції

Розробка даного модуля починається з розробки класу конфігурації контексту додатка Spring. Даний клас має назву SqlInjectionConfig та містить методи, що конструюють об’єкти класів SqlInjectionAnalyzer та PayloadGenerator. Даний клас також містить метод, який створює об’єкт WebDriver для запуску та керування браузером.

```
@Configuration
public class SqlInjectionConfig {
@Bean
public SqlInjectionAnalyzer getSqliAnalyzer() {
return new SqlInjectionAnalyzer();
}
     @Bean
     public List<DBMSSpecificPayload> getPayloads() {
          return Lists.asList(
               new Common(),
               new DBMSSpecificPayload[]{
               new Oracle(),
               new MySql()});
}
@Bean
public PayloadGenerator getSqlInjectionPayloadGenerator()
{
     return new PayloadGenerator(getPayloads());
}
@Bean(destroyMethod = "quit")
@Lazy
```

```
@Qualifier("sqliWebdriver")
@Scope(SCOPE_SINGLETON)
public WebDriver sqliWebdriver()
{
System.setProperty("webdriver.chrome.driver",
SqlInjectionConfig.class.getResource
("/driver/chrome_driver.exe").getPath());
return new ChromeDriver();
}}
```

Тип атаки Boolean blind SQL ін’єкції використовує особливість використання булевих операторів в SQL запитах, а саме використання виразів, що завжди істинні або хибні.

Таким чином, кожний набір даних для атаки складається з двох строк з двома різними булевими виразами на певному діалекті SQL. Для опису такого набору даних розроблений клас BooleanBlindPayload.

Оскільки існує багато діалектів SQL, що залежать від системи керування базами даних, для кращою структуризації та масштабованості запропоновано розділити набори даних до різних класів залежно від діалекту. Це досягається введенням інтерфейсу DBMSSpecificPayload, який використовується для отримання наборів даних незалежно від типу системи керування базами даних.

```
public interface DBMSSpecificPayload
{
List<BooleanBlindPayload>
getDbmsSpecificBlindBasedPayload();
}
```

Класи, які реалізують даний інтерфейс, мають назви Common, MySql та Oracle. Ці класи містять дані для атак загального призначення, специфічні для систем керування базами даних MySQL та Oracle відповідно.

Головним класом модуля є клас SqlInjectionAnalyzer, який відповідає за проведення SQL ін’єкції та аналізу результатів ін’єкції на наявність вразливості.

SQL ін’єкція виконується наступним чином. Для кожного параметра URL формується набір даних для атаки. Спочатку отримується HTML код сторінки без ін’єкції, після чого виконується ін’єкція булевого виразу, який не впливатиме на результуючу вибірку з бази даних і отримується HTML код сторінки після першої ін’єкції SQL. Потім визначається відстань між HTML кодом сторінок за допомогою критерію Джаро – Вінклера.

Після цього виконується ін’єкція булевого виразу, який впливатиме на результуючу вибірку з бази даних або вибираючи усі дані з таблиць, або не вибираючи ждоних, та отримується HTML код сторінки після другої ін’єкції SQL. Потім визначається відстань між HTML кодом сторінки після першої та другої ін’єкцій за допомогою критерію Джаро – Вінклера.

Якщо критерій Джаро – Вінклера менший певного порогового значення, вважається, що сторінка може мати вразливість до SQL ін’єкції.

```
String initialResponse = fetchHtml(url);
String urlWithInjectedNotAlteringPayload =
UrlUtils.appendParameterValue(url, parameterName,
payload.getSameResultPayload());
String nonModifyingInjectedResponse =
fetchHtml(urlWithInjectedNotAlteringPayload);
String urlWithInjectedAlteringPayload =
UrlUtils.appendParameterValue(url, parameterName,
payload.getModifiedResultPayload());
String modifyingInjectedResponse =
fetchHtml(urlWithInjectedAlteringPayload);
return StringUtils.getJaroWinklerDistance
(nonModifyingInjectedResponse, modifyingInjectedResponse)<
jaroWinklerLikenessThreshold;
```

### 5.2.7 Розробка модуля функцій загального призначення

Модуль функцій загального призначення містить класи та процедури, які використовуються або можуть бути використані в декількох модулях та не прив’язані до стану цих модулів. В даній реалізації методу тестування вразливостей Web-додатків є потреба в функціях аналізу та перетворення URL та функціях взаємодії з сервером безпосередньо через протокол HTTP. Для цього розроблено класи UrlUtils та HttpUtils відповідно.

Клас UrlUtils надає наступні статичні методі для взаємодії із URL:

– parseUrl, який перетворює об’єкт типу String на об’єкт типу URL;

– replaceParameterValue, який виконує заміну значення вказаного параметру URL на надане нове значення;

– appendParameterValue, який виконує конкатенацію до значення вказаного параметру URL нового значення;

– ensureAbsolute, який на основі базового URL та часткового відносного URL будує абсолютний URL.

Клас HttpUtils надає можливість відправлення GET запитів до віддаленого серверу за допомогою внутрішнього класу HttpGetRequest. Внутрішній клас HttpGetRequest надає можливість встановлення заголовків HTTP запиту та виконання запиту з отриманням відповіді від серверу у вигляді об’єкту типу String. Клас HttpUtils у даному випадку реалізує шаблон проектування «Фабричний метод».

### 5.2.8 Структура модулів аналізу вразливостей до SQL-ін'єкцій і DOM XSS

Як було вказано вище, модуль аналізу вразливостей *DOM XSS* призначений для аналізу окремої *Web*-сторінки на наявність вразливості

*DOM XSS*. Відповідно до алгоритму тестування вразливості на *DOM XSS*, модуль має класи для отримання *Javascript* коду *Web*-сторінки, класи для аналізу *Javascript*-коду і клас для проведення атак.

Головним класом модуля є клас *DomXssAnalyzer*, який відповідає безпосередньо за аналіз *Web*-сторінки, на яку вказує переданий *URL*, на наявність вразливості *DOM XSS*.

Цей клас компонує усі класи модуля для виконання аналізу і є відповідальним за визначення типу рефлексії *GET*-параметра *URL*, а також безпосередньо за проведення атаки.

Клас *DomScanner* використовується для аналізу початкового *HTML*-коду сторінки і отримання списку усіх тегів *script* з їх вмістом. Клас містить потоковий *HTML*-"парсер" у вигляді класу *ResponseHtmlHandler*, який формує список тегів *script*.

Для виконання аналізу *Javascript*-коду сторінки на наявність маркерів вразливості до *DOM XSS* атак використовуються класи *Javascript* і *JavascriptScanner*. Клас *Javascript* використовується для опису характеристик одиниці *Javascript*-коду *Web*-сторінки, а саме його тип (зовнішній або внутрішній), посилання на зовнішній файл, безпосередньо *Javascript*-код і список маркерів вразливості, який заповнюється після аналізу. Клас *JavascriptScanner* використовується для отримання змісту зовнішніх *Javascript*-файлів і для аналізу *Javascript*-коду на наявність маркерів вразливості у вигляді стоків і витоків.

Клас *PayloadGenerator* використовується для генерації даних для атаки залежно від типу рефлексії параметра *GET* запиту *URL*.

Загальна *UML*-діаграма модуля аналізу вразливостей *DOM XSS* приведена на рис. 5.7.

Модуль аналізу вразливості до *SQL* ін'єкцій призначений для аналізу окремої *Web*-сторінки на наявність вразливості до *SQL*-ін'єкції через *GET* параметр *URL Web*-сторінки.

Головним класом модуля є клас *SqlInjectionAnalyzer*, який відповідає за проведення *Boolean blind SQL* ін'єкції і аналізу результату атаки з метою визначення наявності потенційної вразливості.

Однією зі складнощів тестування вразливості до *SQL*-ін'єкцій є велика кількість діалектів *SQL* залежно від використовуваної системи управління базами даних. Цей факт примушує формувати дані для атаки на декількох діалектах для максимального покриття векторів атаки.

З одного боку, це збільшує кількість вхідних даних імітаційної моделі, підвищує складність проекту і негативно впливає на загальні часові характеристики реалізації і проведення тестування вразливості.

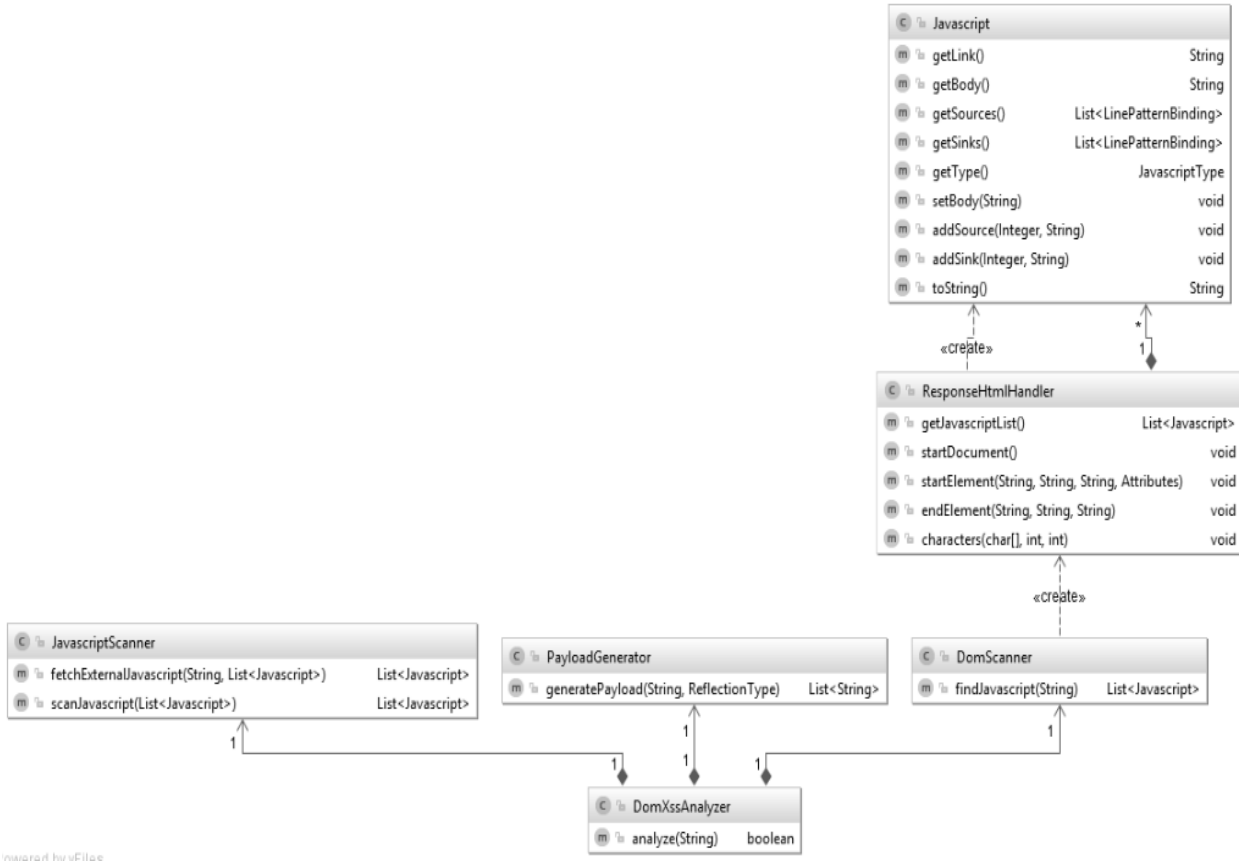


Рис. 5.7. *UML*-діаграма модуля аналізу вразливостей *DOM XSS*

З іншого боку, використовуючи запропонований підхід масштабування, можна істотно знизити складність проекту, не погіршуючи якісних характеристик.

Враховуючи це і з метою усунення дублювання коду, в застосунку реалізований інтерфейс *DBMSSpecificPayload*, який описує метод, що повертає список даних для проведення атаки з урахуванням конкретної системи управління базами даних. Класами, що реалізовують цей інтерфейс, являються *Common, Oracle* і *MySql* для загальних даних атаки, даних, специфічних для систем управління базами даних *Oracle* і *MySQL* відповідно.

Дані для атаки зберігаються в застосунку у вигляді класу *BooleanBlindPayload*, який включає необхідні для проведення атаки дані.

Загальна *UML*-діаграма модуля аналізу вразливості до *SQL*-ін'єкції приведена на рис. 5.8.

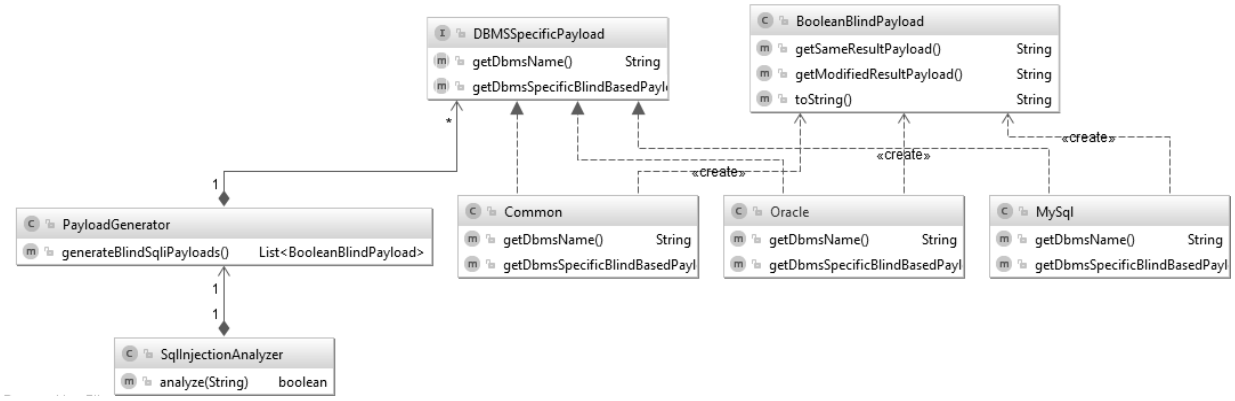


Рис. 5.8. *UML*-діаграма модуля аналізу вразливості до *SQL*-ін'єкції

### 5.2.9 Структура модуля функцій загального призначення

Модуль функцій загального призначення надає набір функцій для роботи з *URL*-посиланнями і їх параметрами, а також для взаємодії з сервером безпосередньо за допомогою протоколу *HTTP*.

Для перетворення *URL* і маніпуляції параметрами *URL* використовується клас *UrlUtils*. Оскільки функції перетворення *URL* і маніпуляції їх параметрами не мають стану, вони реалізовані як статичні методи.

Для взаємодії з сервером по протоколу *HTTP* використовується клас *HttpUtils*. Цей клас реалізує шаблон проектування "Будівельник" для будівництва внутрішнього класу *HttpGetRequest*. Будівельник - твірний шаблон проектування, який дозволяє відокремити конструювання комплексних об'єктів від їх реалізації [23].

В даному випадку, шаблон використаний для конструювання складного об'єкту класу *HttpGetRequest*, який інкапсулює результат виконання *GET* запиту протоколу *HTTP*.

*UML*-діаграма модуля функцій загального призначення зображена на рис. 5.9.

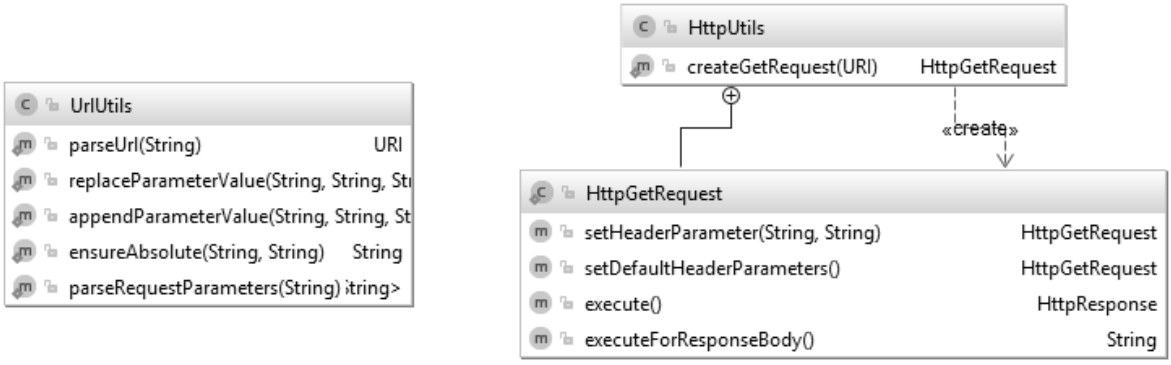


Рис. 5.9. *UML*-діаграма модуля функцій загального призначення

Імітаційна модель технології тестування вразливостей *Web*-застосунків є комплексним програмним забезпеченням з великою кількістю модулів і класів, що у свою чергу, створює проблему компонування класів між собою.

Цієї проблеми можна уникнути зменшенням кількості класів, але це приведе до збільшення сильно пов'язаних між собою класів, що у свою чергу ускладнить розроблення, підтримку, читабельність і масштабованість програмного забезпечення. Для вирішення завдання компонування класів використано механізми "впровадження залежностей" [23].

З літератури [11, 12] відомо, що "впровадження залежностей" - це шаблон проектування, в якому залежності (чи сервіси) впроваджуються, або передаються за посиланням в залежний об'єкт (клієнт) і стають частиною клієнтського стану. Шаблон відділяє створення залежностей клієнта від власної логіки клієнта, дозволяє компонентам бути слабо пов'язаними і дотримуватися принципів інверсії залежностей і єдиного боргу. Існує три основні типи "впровадження залежностей" в клас:

– впровадження в конструктор;
– впровадження у властивість;
– впровадження в метод.

У загальному випадку шаблон "впровадження залежностей" не прив'язаний до програмної реалізації і не вимагає окремих бібліотек або фреймворков для реалізації, проте використання контейнерів для "впровадження залежностей" значно спрощує процес розроблення.

Одним з найпопулярніших контейнерів для "впровадження залежностей" в екосистемі *Java* є *Spring Framework*. *Spring Framework* - фреймворк для застосунків, розроблених на мові програмування *Java*, який надає контейнер для впровадження залежностей, підтримку аспектно-орієнтованого програмування і багато інших функцій [24].

Проведені дослідження показали, що актуальною на момент розроблення є версія 4.3.8 від 18.04.2017, що свідчить про постійний розвиток фреймворка.

Контейнер *Spring* дозволяє централізувати створення об'єктів застосунку і автоматично вирішити усі залежності створених компонентів. При цьому контейнер бере на себе завдання по створенню компонентів, управлінню їх життєвими циклами і знищенню компонентів при знищенні контейнера. *Spring* також підтримує ще один вид "впровадження залежностей", а саме ін'єкцію в приватне поле.

Для вирішення залежностей *Spring* використовує зовнішній конфігураційний файл у вигляді *XML*-файлу або *Java*-класу. Використання контейнера *Spring* має наступні етапи:

– Конфігурація контейнера в зовнішньому файлі.
– Створення контексту застосунку з посиланням на файл конфігурації.
– Отримання необхідних компонентів з контексту.

Для розроблення імітаційної моделі технології тестування *Web*-застосунків було обрано конфігурацію контейнера для "впровадження залежностей" у вигляді *Java*-класу. Кожен модуль багатомодульного проекту містить свій клас з конфігурацією, який відповідає за створення компонентів лише свого модуля, після чого всі конфігураційні файли будуть використані при створенні контексту застосунку, що, у свою чергу, приведе до створення контейнера для "впровадження залежностей" *Spring* і створенню усіх необхідних компонентів застосунку.

Клас конфігурації компонентів *Spring* має певну структуру. Мінімальною вимогою до класу є анотування класу анотацією *@Configuration*, що дозволяє використати клас для конфігурації контейнера для впровадження залежностей при запуску застосунку. Клас конфігурації повинен містити методи, що створюють компоненти застосунку. Ці методи мають бути анотовані анотацією *@Bean*.

При використанні *Spring*, для повного впровадження всіх залежностей усі компоненти мають бути створені в конфігураційних класах. У разі

використання декількох конфігураційних класів, повинен існувати головний конфігураційний клас, анотований окрім анотації *@Configuration* також анотацією *@Import*, що містить посилання всім іншим конфігураційним класам.

Таким чином, отримано подальший розвиток імітаційної моделі технологій тестування безпеки *Web*-застосунків. В основу розроблення покладені основні положення теорії масштабування імітаційних моделей у рамках алгоритмічного спрощення на основі оцінки транзитивної залежності за управлінням і даними.

Відмінною особливістю розробленої імітаційної моделі є адаптація вибору вхідних операторів управління і даних до підвищення вимог оперативності розроблення і реалізації моделі, виражена в реалізації процедури взаємодії з реальним браузером з використанням засобів автоматизації браузеру і формуванні даних для атаки на декількох діалектах.

Такі зміни дозволили знизити обчислювальну складність процедур тестування вразливостей до *SQL*-ін'єкцій і *DOM XSS* до 1,5 разів.

## 5.3 Тестування імітаційної моделі розробленої технології

Опишемо процедуру тестування імітаційної моделі на прикладі аналізу вразливостей *DOM XSS*.

Для тестування імітаційної моделі у відповідному режимі необхідно запустити зібраний проект з параметрами командного рядка *dxss*. Для тестування було обрано *Web*-приложение *https://xss-game.appspot.com/level1*, яке надане фірмою *Google* для тренування навичок інженерів з інформаційної безпеці.

Проведений аналіз показав, що цей *Web*-застосунок створений з відомою вразливістю до *DOM XSS* атак. Розроблений застосунок в режимі тестування має наступні параметри командного рядка :

*dxss https://xss - game.appspot.com/level1/frame?query=123*

Після запуску застосунку відкривається вікно браузеру з відповідним *URL* (рис. 5.10).

Після відкриття *Web*-сторінки виконується аналіз *Javascript*-коду на сторінці. При аналізі в журнал застосунку виводиться інформація про знайдені *Javascript*-файли і їх типи (рис. 5.11).

Після аналізу *Javascript*-коду на наявність маркерів вразливості виконується визначення типу рефлексії параметра *URL* (рис. 5.12). Для цього генерується унікальне значення і вставляється як значення параметра *URL*, після чого виконується перехід за посиланням.


Рис. 5.10. Зовнішній вигляд вікна браузеру з відповідним *URL*

```
[00:51:58] INFO: Fetching HTML contents from https://xss-game.appspot.com/level1/frame?query=123
[00:55:32] INFO: Javascript /static/game-frame.js is external. Fetching external JS.
[00:55:46] INFO: Found 1 javascript tags in page. Starting analysis for possible DOM XSS sources and sinks
[00:55:49] INFO: Scan complete. Found 1 javascript tags with possible XSS vulnerabilities
```

Рис. 5.11. Зовнішній вигляд інформації про знайдені *Javascript*-файли та їх типи


Рис. 5.12. Зовнішній вигляд вікна визначення типу рефлексії параметра *URL*

Потім, виходячи з типу рефлексії параметра, визначається набір даних для атаки і проводиться атака з кожним набором даних.

Для простоти тестування в якості маркера було використано діалогове вікно *Alert*. У разі успішної атаки (рис. 5.13) набір даних записується (логується) у текстовий файл.

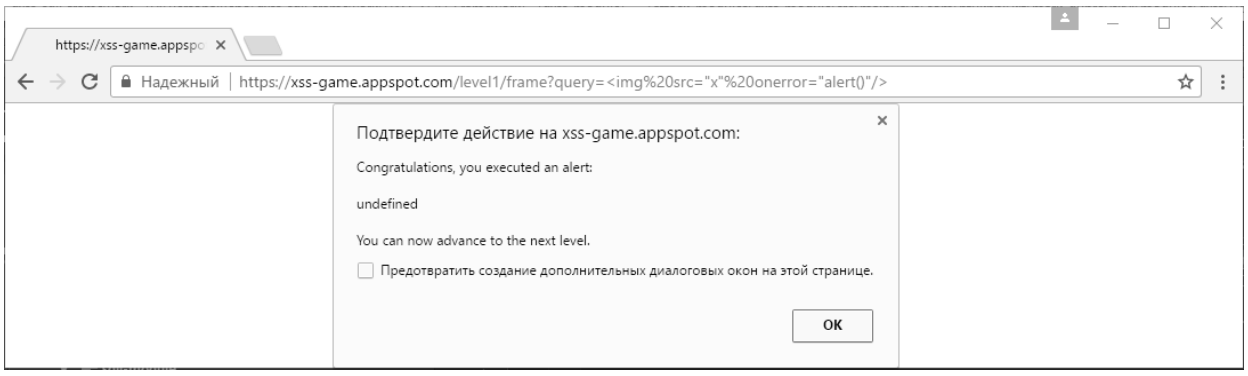


Рис. 5.13. Зовнішній вигляд вікна, що сигналізує про успішно проведену *DOM XSS* атаку

## 5.4 Висновки до розділу

У розділі отримано подальший розвиток імітаційної моделі технології тестування безпеки на основі положень теорії масштабування імітаційних моделей. Відмінною особливістю розробленої імітаційної моделі є адаптація вибору вхідних операторів управління і даних до підвищення вимог оперативності розроблення і реалізації моделі, виражена в реалізації процедури взаємодії з реальним браузером з використанням засобів автоматизації браузеру і формуванні даних для атаки на декількох діалектах.

В основу запропонованого підходу алгоритмічного спрощення імітаційного моделювання покладено вдосконалені процедури оцінки транзитивної залежності за управлінням і даними.

Визначено допустимість і доцільність використання оцінки транзитивної залежності, що знизить обчислювальну складність реалізованих алгоритмів в порівнянні з алгоритмами оцінки прямої залежності до 1,5 разів.

Отпубліковані наукові результати роботи автора приведені у 5 розділі [6, 7, 8, 25, 26].

**Список використаних джерел:**

[1] Сирота А.А. Компьютерное моделирование и оценка эффективности сложных систем / А.А. Сирота. –М.: Техносфера, 2006. –280 с.

[2] Макконнелл С. Совершенный код. Мастер-класс. М.: Рус. Редакция ; СПб.: Питер, 2007. 896 с.

[3] Machine Learning Techniques for Anomaly Detection// [ Salima Omar, Asri Ngadi, Hamid H. Jebur ]// International Journal of Computer Applications (0975 – 8887).– 2013, vol.79, No.2.– P. 33-41.

[4] Markus Goldstein. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data./ Markus Goldstein andSeiichi Uchida// PLoS ONE. – 2016, vol. 11, no. 4, P.28-32.

[5] Patcha, A. An overview of anomaly detection techniques: Existing solutions and latest technological trends / A. Patcha, J. M. Park : Computer Networks. – 2007. – Vol. 51, №. 12. – pp. 3448-3470.

[6] Коваленко А.В. Масштабирование имитационной модели технологии тестирования безопасности / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (46). – Полтава: ПолтНТУ. – 2017. – С. 181-184.

[7] Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

[8] Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

[9] Seacord Robert The CERT C Secure Coding Standard / Robert C. Seacord – Addison-Wesley, 2008. - pp 720.

[10] Takagi T. Fuzzy identification of systems and its applications to modeling and control / T. Takagi, M. Sugeno // IEEE Transactions on Systems, Man and Cybernetics. – 1985. – Vol. SMC-15, No. 1. – P. 116–132.

[11] McConnell S. Software Estimation: Demystifying the Black Art (Developer Best Practices). Microsoft Press, 2006. 308 p.

[12] Семенов С.Г., Лисица Д.А. Модель оценки риска разработки программного обеспечения: сб. материалов XV Международной НТК «Проблемы информатики и моделирования». Харьков: НТУ «ХПИ»,

2015. С.82.

[13] Koller D. and N.Friedman, Probabilistic Graphical Models. Principles and Techniques. Massachusetts: MIT Press, 2009. – 1280 p.

[14] Липаев В.В. Надежность и функциональная безопасность комплексов программ реального времени. Москва, 2013. 176 с.

[15] НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999р., №22. [Електронний ресурс]. – Режим доступу до ресурсу: http://do.gendocs.ru/docs/index-27508.html?page=8

[16] Смірнов С.А. Метод антивірусного захисту даних з використанням хмарних обчислювальних технологій дис. кандидата техн. наук: 05.13.21 [Текст] / Смірнов Сергій Анатолійович. – Київ, 2017. – 128 с.

[17] Шибанов, А.П. Обобщенные GERT-сети для моделирования протоколов, алгоритмов и программ телекоммуникационных систем: дис. доктора техн. наук: 05.13.13 [Текст] / Шибанов Александр Петрович. – Рязань, 2003. – 307 с.

[18] Ruby Sam Agile Web Development with Rails / Sam Ruby, Dave Thomas, David Heinemeier Hansson – The Pragmatic Bookshelf, 2011. – 480 с.

[19] Hasan Yasar Security Practitioner Perspective on DevOps for Building Secure Solutions / Режим доступу: http://www.sei.cmu.edu/webinars/ view_webinar.cfm?webinarid= 474101&gaWebinar= SecurityPractitioner PerspectiveonDevOpsforBuildingSecureSolutions

[20] Maven – Introduction: [Електронний ресурс]. – Режим доступу: https://maven.apache.org/what-is-maven.html

[21] Maven – POM Reference: [Електронний ресурс]. – Режим доступу: https://maven.apache.org/pom.html

[22] Постанова Кабінета Міністрів України від 29.03.2006 №373 «Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах» [Електронний ресурс]. – Режим доступу до ресурсу: http://zakon2.rada.gov.ua/laws/show/373-2006-п

[23] Девятов С.С. Проектирование программного обеспечения с использованием стандартов UML 2. 0 и SYSML 1.0 / С.С. Девятов // Прикладная информатика. – М.: Негосударственное образовательное частное учреждение высшего образования Московский финансово-промышленный университет Синергия. – 2006. – №6 2– С. 48-63.

[24] Spring Framework: [Електронний ресурс]. – Режим доступу: http://projects.spring.io/spring-framework/

[25] Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL-ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

[26] Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп’ютерних систем для підвищення безпеки даних: монография / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 350 с.

# РОЗДІЛ 6
# МЕТОД ПЕРЕДТЕСТОВОЇ КОМПІЛЯЦІЇ І РОЗПОДІЛУ ДОСТУПУ. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНИХ МЕТОДІВ І ОБҐРУНТУВАННЯ ПРАКТИЧНИХ РЕКОМЕНДАЦІЙ З ЇХ ВИКОРИСТАННЯ

У розділі проведено дослідження ефективності розроблених моделей і методів тестування безпеки застосунків, оцінку достовірності отриманих результатів математичного моделювання, а також обґрунтування практичних рекомендацій з використання методів та засобів управління безпекою.

## 6.1 Порівняльні дослідження і оцінка ефективності розроблених моделей і методів тестування безпеки

На основі проведеного імітаційного моделювання процесу тестування безпеки застосунків оцінимо ефективність розроблених моделей і методів тестування порівняно з існуючими засобами і технологіями, заснованими на відомому алгоритмі Пурдома [1, 2].

Практичний досвід тестування безпеки застосунків показує, що визначення точних кількісних даних вразливостей застосунків, як правило, не представляється можливим в силу великої кількісті невизначеностей вхідних даних. Тому широке поширення отримали наближені оцінки, засновані на обробці емпіричних даних, зібраних в процесі тестування безпеки.

У дисертаційній роботі в якості основи для оцінки ефективності розробленої технології тестування безпеки застосунків використано один з методів статистичного аналізу - зведення і угрупування статистичних даних, що отримав теоретичне обґрунтування в роботах [3, 4].

При проведенні дослідження було взято 20 *Web*-застосунків з різною кількістю (від 31 до 77) тестованих елементів.

В результаті експериментів отримано значення часу тестування безпеки застосунків для способів, використовуючих алгоритм Пурдома, і розроблену технологію тестування безпеки. Результати тестування представлені в табл. 6.1.

Таблиця 6.1. Результати тестування безпеки *Web*-застосунків

| N | Час тестування (алг. Пурдома) (с.) | Час тестування (розроблена технологія тестування) (с.) | N | Час тестування (алг. Пурдома) (с.) | Час тестування (розроблена технологія тестування) (с.) |
|---|---|---|---|---|---|
| 1 | 23 | 22,4 | 11 | 14,7 | 14,5 |
| 2 | 15,3 | 14,5 | 12 | 29,2 | 28,2 |
| 3 | 44 | 41,1 | 13 | 45,6 | 44,1 |
| 4 | 23,5 | 22,1 | 14 | 11,1 | 11 |
| 5 | 43,7 | 41,0 | 15 | 19,3 | 18,6 |
| 6 | 24,1 | 22,7 | 16 | 12,4 | 12 |
| 7 | 33 | 31,6 | 17 | 31,6 | 30,2 |
| 8 | 52 | 48,7 | 18 | 20,1 | 19.6 |
| 9 | 17,8 | 17,4 | 19 | 33,1 | 31,4 |
| 10 | 20,1 | 19,3 | 20 | 24 | 22,9 |

Побудуємо інтервальний ряд розподілу часу тестування безпеки, для чого оберемо оптимальний інтервал $k$ і встановимо розмах інтервалу $h$.

Оптимальне число груп оберемо так, щоб в достатній мірі відбилася різноманітність значень ознаки в сукупності, і в той же час закономірність розподілу, його форма не спотворювалася випадковими коливаннями частот, при цьому скористаємося формулою Стерждесса [5, 6].

У нашому випадку оптимальний інтервал: *k = 1 + 3,322lg20 = 5,32*. Оскільки число груп не може бути дробовим, то округлюємо *k = 5,32* до найближчого цілого числа за правилами округлень - *5*.

Знаючи число груп, розрахуємо довжину (розмах) інтервалу за формулою:

$$h = (X_{max} - X_{min}) / k$$

Виходячи з даних, визначених вище:

$$h1 = (52 - 11{,}1) / 5 = 8{,}18,\ h2 = (48{,}7 - 11) / 5 = 7{,}54$$

Таким чином, інтервальний ряд розподілу часу тестування безпеки розіб'ємо на 5 груп з інтервалом по 8,18 с. і 7,68 с. Представимо інтервальний ряд розподілу тестування безпеки застосунків у вигляді табл. 6.2.

Таблиця 6.2. Інтервальний ряд розподілу тестування безпеки *Web*-застосунків

| Час (алг. Пурдома), с. | Число потраплянь в інтервал | Час (розроблена технологія тестування), с. | Число потраплянь в інтервал |
|---|---|---|---|
| 11,1-19,28 | 5 | 11-18,54 | 5 |
| 19,28-27,46 | 7 | 18,54-26,08 | 7 |
| 27,46-35,64 | 4 | 26,08-33,62 | 4 |
| 35,64-43,82 | 1 | 33,62-41,16 | 2 |
| 43,82-52 | 3 | 41,16-48,7 | 2 |

Як видно з представленої таблиці, навіть така невелика вибірка тестованих застосунків показала переваги розробленої технології тестування безпеки.

Так, максимальне значення інтервального ряду зменшилося при використанні розроблення в 1,07 разу, зменшилося число попадань в максимальний часовий інтервал, а також сумарний час тестування зменшився в 1,05 разу.

Слід зауважити, що на практиці найчастіше не використовується теоретично обґрунтований алгоритм Пурдома. При цьому тестування безпеки проводиться виходячи з досвіду тестувальників. В цьому випадку розроблена технологія тестування має істотну перевагу (до 1,5 разу).

Результати тестування безпеки *Web*-застосунків візьмемо для оцінки ефективності синтезованих моделей та методів розроблення безпечного ПЗ. При цьому для практичного прикладу використання цих моделей та методів візьмемо за основу комп'ютерну систему управління інформаційними потоками авіатранспортного підприємства (КСУ ІП АП).

Структурна схема КСУ ІП АП наведена на рис. 6.1.

Як видно з цього рисунку, для вирішення завдань управління інформаційними потоками в наведеній КС вже використовуються різні методи і засоби, реалізовані в загальну систему управління авіатранспортним підприємством.

Вони адаптовані під сучасні операційні системи та технології розгортання ПЗ, що дає можливість використовувати механізми комунікації, маршрутизації, прогнозування та управління навантаженням комп'ютерних мереж та ін.

Для оцінки ефективності синтезованих моделей та методів використамо наведений у розділі 1 показник безпеки ПЗ $\overline{Y_i^{(\text{ПЗ})}}$ (вираз 1.1) та його векторні складові $\overline{Y_{mat}^{(\text{ПЗ})}}$, $\overline{Y_{fft}^{(\text{ПЗ})}}$, $\overline{Y_{rec}^{(\text{ПЗ})}}$, $\overline{Y_{conft}^{(\text{ПЗ})}}$, $\overline{Y_{intt}^{(\text{ПЗ})}}$, $\overline{Y_{autht}^{(\text{ПЗ})}}$, $\overline{Y_{avb}^{(\text{ПЗ})}}$, $\overline{Y_{fir}^{(\text{ПЗ})}}$.

У табл. 6.3. наведено результати порівняльних досліджень показників, що мають вплив на векторні показників безпеки ПЗ зокрема, та загальний стан безпеки ПЗ КС в цілому.

Як видно з табл. 6.3. використання синтезованих моделей та методів розроблення безпечного ПЗ дозволить покращити показники глибини тестового контролю безпеки ПЗ $K_{test\ depth}$ до 5-7%, ймовірності помилки $P_{software\ error}$ до 5%, ймовірності блокування доступу до ресурсів $P_{blocking}$ до 30%.

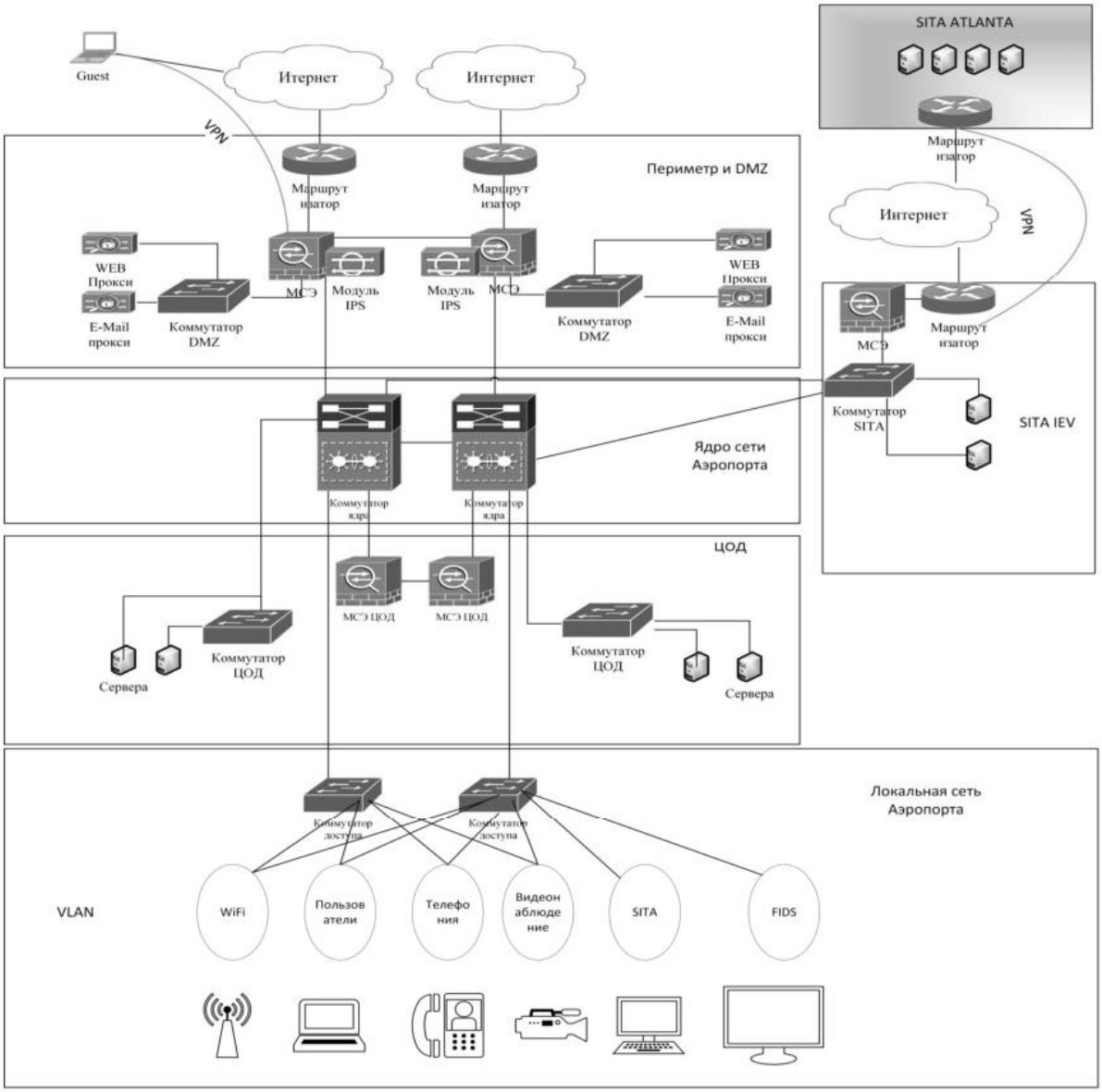


Рис. 6.1. Структурна схема КСУ ІП АП

Покращення досліджуваних показників, в свою чергу, дозволить покращити вказані в табл. 6.3. векторні показники. Використання виразу 1.1. дозволило визначити показник безпеки ПЗ $Y_i^{(ПЗ)}$ та розрахувати його для наведеного прикладу КС. Проведені дослідження показали, що показник $Y_i^{(ПЗ)}$ збільшився до 15%.

Таблиця 6.3. Результати порівняльних досліджень показників, що мають вплив на векторні показників безпеки ПЗ

| № з.п | Досліджуваний показник | Значення прі використанні розроблених моделей та методів | Значення прі використанні існуючих систем | Векторні складові показники безпеки ПЗ |
|---|---|---|---|---|
| 1 | Глибина тестового контролю безпеки ПЗ $K_{test\ depth}$ | 90-97% | 85-90% | $\overline{Y_{mat}^{(ПЗ)}}$, $\overline{Y_{fft}^{(ПЗ)}}$, $\overline{Y_{rec}^{(ПЗ)}}$, $\overline{Y_{conft}^{(ПЗ)}}$, $\overline{Y_{intt}^{(ПЗ)}}$, $\overline{Y_{autht}^{(ПЗ)}}$, $\overline{Y_{avb}^{(ПЗ)}}$ |
| 2 | Ймовірність помилки $P_{software\ error}$ | 0,9-0,95 | 0,8-0,9 | $\overline{Y_{fft}^{(ПЗ)}}$, $\overline{Y_{avb}^{(ПЗ)}}$, $\overline{Y_{rec}^{(ПЗ)}}$, $\overline{Y_{conft}^{(ПЗ)}}$, $\overline{Y_{intt}^{(ПЗ)}}$, $\overline{Y_{autht}^{(ПЗ)}}$ |
| 3 | Ймовірність блокування доступу до ресурсів $P_{blocking}$ | 0,8-0,95 | 0,6-0,85 | $\overline{Y_{fft}^{(ПЗ)}}$, $\overline{Y_{rec}^{(ПЗ)}}$, $\overline{Y_{intt}^{(ПЗ)}}$, $\overline{Y_{avb}^{(ПЗ)}}$ |

Таким чином, проведені дослідження та експертна оцінка дозволили зробити висновок, що використання синтезованих моделей та методів розробки безпечного ПЗ дозволить підвищити рівень захисту інформації в існуючих КС.

## 6.2 Обґрунтування достовірності отриманих результатів моделювання

Перевірку достовірності отриманих результатів в четвертому та п’ятому розділах здійснено за допомогою експериментів, згідно зазначених умов:

– група тестувальників складається з 3 чоловік, з них два тестувальники безпеки і 1 один *Person Non* [7, 8];

– основна методологія управління розробкою є *SCRUM*;

– число експериментів *N* = 20*.

За результатами експерименту отримано гістограму часу тестування безпеки застосунків [9, 10], представлена на рис. 6.2.

Перевіремо за критерієм згоди $\chi^2$ Пірсона [5] припущення про нормальний розподіл цієї випадкової величини:

$$\chi^2 = N\sum_{i=1}^{k}\frac{(P_i^{emp} - P_i^{theor})^2}{P_i^{theor}},$$

де *k* – число інтервалів статистичного ряду; $P_i^{emp}$ і $P_i^{theor}$ – емпіричне і теоретична ймовірності "потроплання" заданого показника в *i*-й розряд.

У результаті перевірка показала що величина часу тестування безпеки програмного продукту розподілена за нормальним законом, тобто підтверджено правдоподібність гіпотези. Проведено обчисленя оцінки математичного сподівання $\bar{t}_{ts}^{(i)}$. Розраховано дисперсіїю $\overline{D}_{t_{ts}^{(i)}}$ випадкової величини часу тестування безпеки програмного проекту що розглядається ($t_{ts}^{(i)}$). Виділено середньоквадратичного відхилення $\overline{\theta}_{t_{ts}^{(i)}}$.

$$\bar{t}_{ts}^{(i)} = \frac{\sum_{i=1}^{k}\bar{t}_{ts}^{(i)}}{N^*},\ \overline{D}_{t_{ts}^{(i)}} = \frac{\sum_{i=1}^{k}\left(\bar{t}_{ts}^{(i)} - \bar{t}_{ts}^{(i)}\right)^2}{N^*\text{-}1},\ \overline{\theta}_{\bar{t}_{ts}^{(i)}} = \sqrt{\overline{D}_{t_{ts}^{(i)}}}.$$

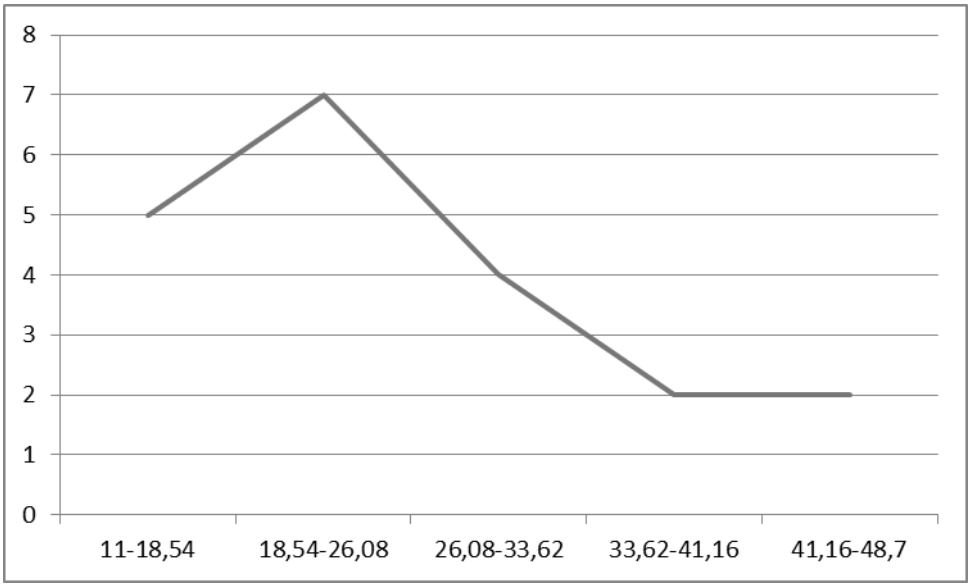


Рис. 6.2. Гістограма часу тестування безпеки програмних продуктів

Розрахуемо дані довірчої ймовірності результату який приведе до того що отримані в результаті експерименту часові дані тестування безпеки програмного проекту «не відхиляться» більше, ніж на одиницю від математичного сподівання $\bar{t}_{ts}^{(i)}$. Для цього використаємо вираз для розрахунку даних в незалежних випробуваннях [5] довірчої ймовірності відхилення відносної частоти від постійної ймовірності.

$$P\left(\left|\bar{t}_{ts}^{(i)} - t_{ts}^{(i)}\right| < 1\right) = 2\Phi\left(\frac{1}{\bar{t}_{ts}^{(i)}}\right), \Phi(x) = \frac{1}{\sqrt{2\pi}}\int_0^x e^{-\frac{z^2}{2}}dt$$

де $\Phi$ – функція Лапласа.

Таким чином після проведених експериментів результат показав, що статистична величина $t_{ts}^{(i)}$ довірчої ймовірність відхилення для видів даних що розглядаються складає $P \approx 0{,}92$, тобто значення статистичної величини від математичного сподівання $\bar{t}_{ts}^{(i)}$ «не відхилиться» більше, ніж на одиницю.

Проведемо порівняльний розрахунок за даними математичного моделювання та експерименту п’ятого розділу по аналогії з вищерозглянутими розрахунками. Результат наведено на рис. 6.3. де

зображено графік часу тестування безпеки $t_{ts}$ програмних продуктів, а саме щільності розподілу ймовірності та меж довірчого інтервалу $I_\beta = \left[\bar{J} - \varepsilon_\beta, \bar{J} + \varepsilon_\beta\right]$. де як можна побачити що довірча ймовірність $\beta = 0{,}94$ отриманих істинних значень $\bar{J}$.

Крім того з графіку видно що отримана суцільна крива $J$ в основній тестовій ситуації потрапляє у заштриховану область – «усереднений» довірчий інтервал даних.

Проведені розрахунки доводять достовірність розроблених у четвертому розділі результатів математичної моделі та результатів аналітичного дослідження.

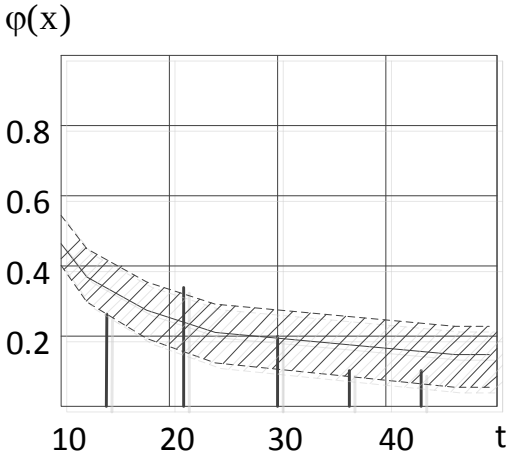


Рис. 6.3. графік часу тестування безпеки програмних продуктів $t_{ts}$ з зазначенням щільності розподілу ймовірності, меж довірчого інтервалу

## 6.3 Розроблення методу передтестової компіляції і розподілу доступу

Проведені дослідження показали, що в процесі управління розробкою ПЗ, проектування, реалізації, верифікації, а також тестування безпеки ПЗ

існує ряд загальних і спеціальних рекомендацій, розроблених і випробуваних на практиці експертами [11-18].

Серед них можна виділити наступні: управління якістю, інжиніринг вимог безпеки, моделювання загроз, аналіз атак, аналіз вразливостей у наявному коді, перевірка вхідних даних, забезпечення безпеки компілятором, статичний аналіз, проникаюче тестування, аудит коду, розроблення керівництва і контрольних списків розробників, незалежний огляд безпеки та ін.

Аналіз представлених рекомендацій показав, що у більшості своїй вони зачіпають відомі розробникам ситуації безпеки. В той же час, вони не враховують можливостей безпосередньої тестової роботи кодерів-розробників ПЗ у рамках методів "Smoke"-тестування і передтестової роботи у рамках компіляції програмного коду.

В той же час, як показують дослідження [7, 9, 10, 19-22], саме у рамках передтестової роботи існують додаткові можливості підвищення безпеки ПЗ за рахунок врахування профілю програм і користувачів, а також оптимізації системи розподілу доступу до досліджуваних даних.

У рамках розроблюваного методу розподілу доступу при оптимізації з урахуванням профілю передкомпіляції програми проводиться необхідний збір даних, що формуються у множині профілів користувачів.

Для підвищення точності врахування профілів користувача, специфіки його діяльності і характеристик комп'ютерної системи пропонується розбиття процесу компіляції на дві фази:

– фаза синтезу ПЗ з урахуванням можливостей сучасних компіляторів;

– фаза адаптації і розподілу доступу до ПЗ з урахуванням профілів програми і користувача.

Таке розділення передтестової компіляції на дві фази дозволить вирішити наступні завдання:

1. Розподіл доступу користувачів з урахуванням можливостей персоналізації відповідних профілів. При цьому враховуються можливості першої фази компіляції - можливості динамічної машинно-незалежної оптимізації на даних конкретного ПЗ і користувача.

2. Врахування внутрішніх характеристик комп'ютерної системи користувачів (архітектури, планувальника команд, та ін.).

3. Врахування можливостей розподілу доступу при збиранню і підтримці ПЗ.

Для вирішення завдань динамічної машинно-незалежної оптимізації доцільно скористатися відомою технологією компіляції *LLVM*.

З літератури [23, 24] відомо, що у рамках цієї технології представлені статичний компілятор, компонувальник, віртуальна машина, *JIT*-компілятор. Функціонування системи забезпечується єдиною внутрішньою структурою, яка може бути проілюстрована в текстовому форматі, у формі структур даних в оперативній пам'яті, а також в двійковому виді як біт-код. Цей біт-код може бути збережений в проміжних об'єктних файлах для подальшої оптимізації, у тому числі динамічної.

При цьому можливо використати усі надані *LLVM* можливості по обробці внутрішнього представлення (включаючи різні аналізи, трансформації і т.п.). Тому інфраструктура *LLVM* надає зручну базу для досліджень з динамічної оптимізації програм [25, 26].

У розроблюваному методі передтестової компіляції і розподілу доступу в першій фазі виконується процедура машинно-незалежної компіляції з використанням *LLVM*. Результат першої фази зберігається у файл *LLVM* і додатково генеруються дані про архітектуру програмного засобу та алгоритм можливої інсталяції.

Виконання другої фази можливе з використанням програмних засобів віртуального моделювання (віртуальних машин), а так само безпосередньо на комп'ютерних системах користувачів з урахуванням особливості їх профілів і характеристик обчислювальних засобів.

Слід зауважити, що в другій фазі методу передтестової компіляції і розподілу доступу можливі декілька варіантів реалізації:

1) Автоматична компіляція з урахуванням налагодженої (представленої) архітектури КС.

2) Оптимізація і інсталяція ПЗ з урахуванням профілю поведінки користувача.

3) Оптимізація та інсталяція з урахуванням профілю поведінки користувача на етапі збирання, підтримки (вимушеного простою).

Структурна схема розроблюваного методу представлена на рис. 6.4.

Слід зауважити, що представлені на рис. 6.4 етапи і додаткові програмні інструменти передбачені з урахуванням використовуваної технології *LLVM*.

В ході першої фази розроблюваного методу передтестової компіляції і розподілу доступу пропонується використання технології *proxy*-компіляції з викликом спеціалізованої утиліти *make.sh*, а також передачі управління і усіх параметрів, використовуваних при формуванні профілю поведінки користувача і розподілу доступу.

При цьому додатково прописуються команди компілятора *LLVM-GCC*.

Для підтримки незалежності процесу збирання ПЗ від скриптових застосунків передбачені процедури збереження початкових результатів. Кінцевим результатом першої фази є інсталяційний пакет зі спеціальними політиками скриптів компіляції і збирання.

Як було вказано вище, у другій фазі можливі декілька варіантів реалізації на основі статичної і динамічної компіляції (*JIT*-компілятор) і інсталяції.

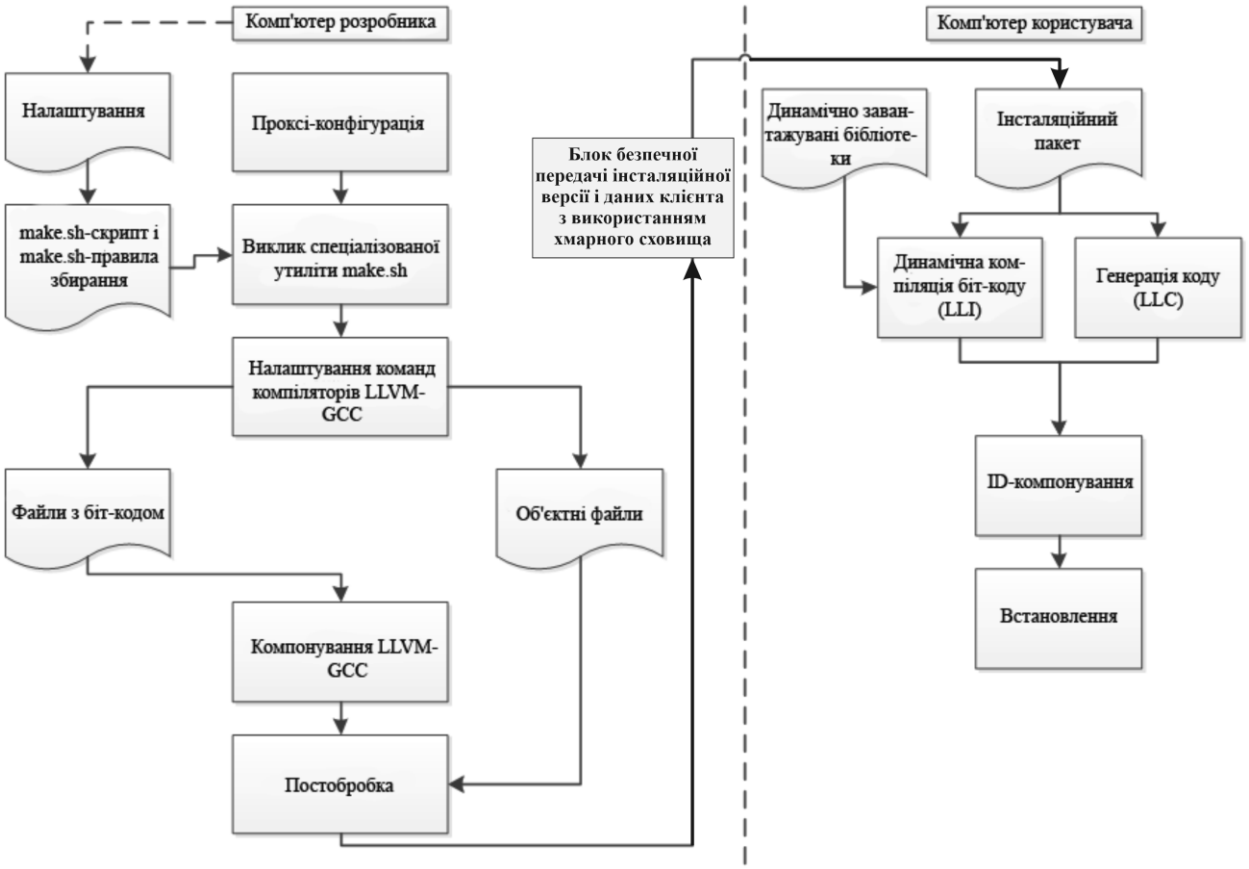


Рис. 6.4. Структурна схема методу передтестової компіляції і розподілу доступу

У обох випадках ці процедури виконуються з урахуванням профілю поведінки користувача або особливостей комп'ютерної системи. У роботі пропонується наступна схема синтезу профілю поведінки користувача (рис. 6.5.).

1) *JIT*-компілятор отримує з інсталяційного файлу біт-код, компілює його і запускає, а також ініціює процес обміну статистичними даними. Модуль ядра операційної системи виконує збирання статистики про особливості комп'ютерної системи і скидає дані статистики у буфер пам'яті.

2) "Демон" профілю користувача має можливість доступу до даних і буфера, розпізнає і записує їх в спеціалізовані канали передачі.

Оскільки код програми, згенерованої в першій фазі, не видно системі і виконується машинно-незалежним компілятором, він може бути поміщений в незалежні простори комп'ютерної системи.

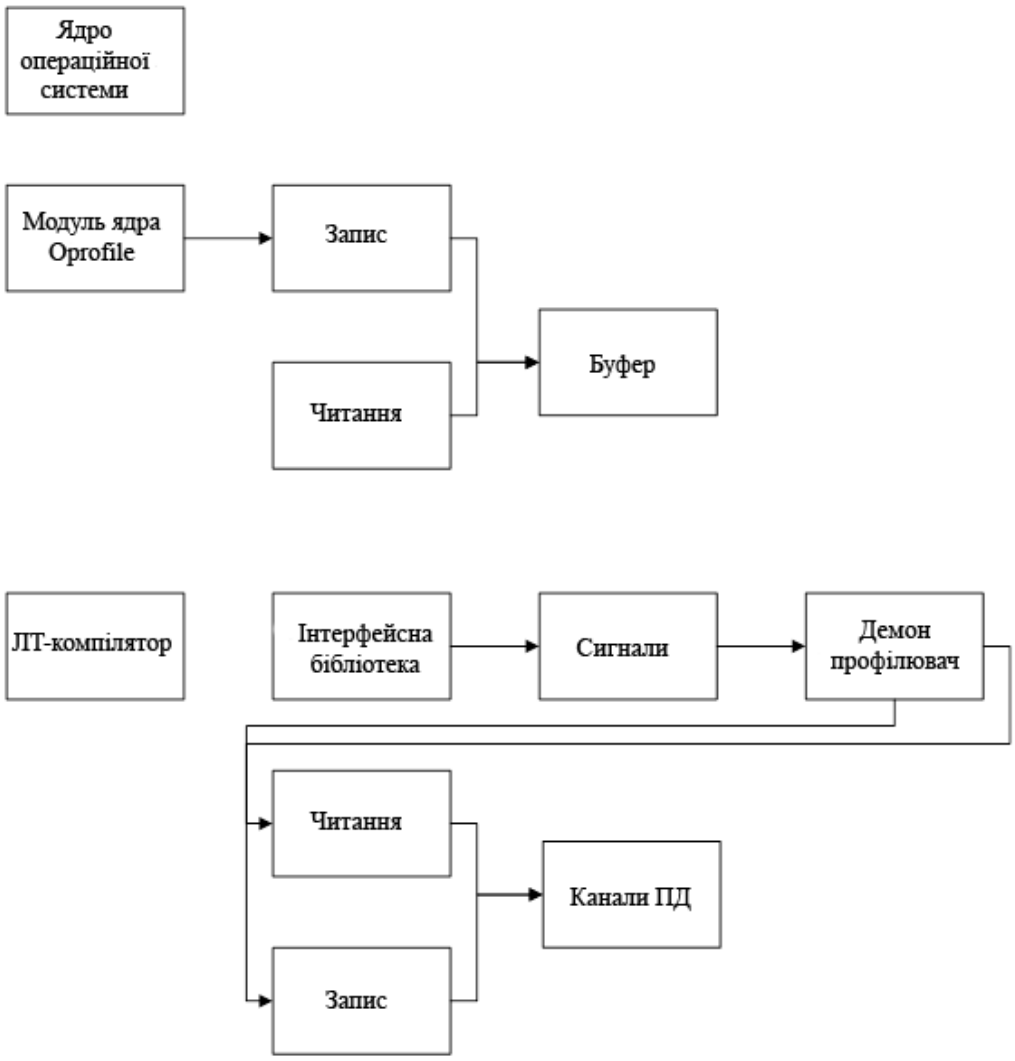


Рис. 6.5. Схема синтезу профілю поведінки користувача

"Демон" вибирає файл з прапором "незалежні дані", записує цю інформацію в канал ПД і продовжує працю до отримання відповідної команди про завершення роботи.

Для коректного використання зібраної інформації про конфігурацію і поведінку користувача під час компіляції потрібне підстроювання отриманого динамічного профілю до точного профілю по базових блоках і ребрах графа потоку управління програми.

Оскільки дані профілю отримані шляхом синтезу окремих даних, то в них неминуче є погрішності, що виникають із-за періоду їх отримання, а також затримок на передачу даних компілятору.

Отже, такі дані не можуть бути безпосередньо використані оптимізаційними фазами *LLVM* таким само чином, як і дані статичного профілю - наприклад, побудований профіль по ребрах графа потоку управління не задовольнятиме рівнянням потоку.

Для вирішення цього завдання можна використати алгоритм, описаний у роботі [22] фахівців *Google* для компілятора *GCC*, в яких автори пропонують підходи модифікації профілю користувача з урахуванням обмежень завдання про максимальний потік мінімальної вартості.

Основні блоки алгоритму представлені на рис. 6.6.

Алгоритм приймає на вхід структурований по базових блоках профіль користувача, а також інформацію статичного аналізу циклів: вона буде потрібна для попередньої оцінки вагів ребер графа потоку управління функції.

Оскільки профіль користувача синтезований для інструкцій, сумарна кількість блоків даних для профілю на базовому блоці оцінима як середнє по частотах інструкцій:

$$S = \frac{\sum_{i=1}^{N} NB_i}{N},$$

де $N$ – число інструкцій у базовому блоці, $NB_i$ – число блоків даних для профілю для $i$ -ої інструкції.


Рис. 6.6. Основні блоки алгоритму переоцінки профілю поведінки користувача

Одним з варіантів рішення поставленої задачі є використання алгоритму Голдберга-Рао [21], що є модифікацією алгоритму Форда-Фалкерсона [28].

Цей досить сучасний (1997 р.) алгоритм уперше поліпшив оцінку *O(nm)*, що вважалася непереборною. Його результат: *O(min{n2/3, n1/2}m Log(n2/m) LogU)*.

Алгоритм є слабо-поліноміальним, але, враховуючи теорему подібності Габова [5], у якій для порівняння слабо- і сильно-поліноміальних алгоритмів використовується *LogU = O(Log n)*, серед фахівців він знайшов позитивну оцінку.

У кожній своїй ітерації алгоритм Голдберга-Рао шукає тупиковий потік, використовуючи неодиничну функцію для визначення довжини допустимих дуг.

Після побудови максимального потоку будується залишкова мережа, в якій по алгоритму, представленому в роботах [10, 24] віддаляються цикли негативної вартості.

По мережі максимального потоку мінімальної вартості відновлюється початковий граф потоку управління: по ребрах, побудованих в результаті перетворення вершин, ми відновлюємо профіль вершин. Ребра, що відповідають початковому графові, не видаляються алгоритмом, оскільки їм призначена нескінченна максимальна пропускна спроможність, і отже, вони теж можуть бути відновлені.

Слід зауважити, що побудовані лічильники можна використати у будь-яких оптимізаціях, що підтримують профіль, як результати збирання звичайного профілю способом інструментування.

Проте, безпосереднє застосування профілю до наявного біт-коду призведе до помилок невідповідності профілю - це обумовлено тим, що перед запуском кодогенерації, *LLVM* додатково виконує декілька проходів, що змінюють машинно-незалежне представлення, і зібраний профіль вже не відповідає збереженому біт-коду.

Таким чином, для практичного застосування профілю поведінки користувача, зібраного шляхом синтезу блоків даних, необхідно виконувати правила і алгоритми, описані вище, а також процедури оптимізації, що виконуються над машинно-незалежним представленням.

Метод передтестової компіляції і розподілу доступу дозволяє виконувати процедури оптимізації на призначеній для користувача машині як динамічно, через створення відповідних *JIT*-компіляторів, так і під час

простою програми, з урахуванням зібраного профілю, що відбиває поведінку користувача, і характеристик його машини.

Проте, в сучасних умовах використання мобільних засобів, часто оптимізація програм на пристрої є скрутною (бракує необхідної кількості ресурсів). Вирішити це протиріччя можна, компілюючи на сервері застосунків, при цьому на пристрій покласти лише завдання збирання і передачі профілю на сервер.

Для організації такого сервера вимагається вдосконалення методу безпечної передачі застосунків. У роботі завдання передачі і обробки застосунків пропонується вирішити за допомогою методу безпечної маршрутизації метаданих в хмарні антивірусні системи, описаного в роботі Смірнова С. А. "Метод антивірусного захисту даних з використанням хмарних обчислювальних технологій" [30].

У цій роботі передачу даних пропонується організувати безпечним шляхом в "хмарне сховище" застосунків, що забезпечує як переносимість програм у рамках одного сімейства процесорної архітектури *ARM*, так і високу міру безпеки застосунків, що зберігаються.

Після передачі і розподілу застосунку в "хмарне сховище" для забезпечення безпеки пакет може бути перевірений інструментами середовища *Svace* на наявність критичних помилок і вразливостей (використовуючи біт-код *LLVM*, а також інформацію про зв'язки програми), а також проведений тест безпеки методами, використаними в середовищі *Trex*.

Таким чином, структурна схема системи передачі і розподілу застосунків у "хмарному сховищі" представлена на рис. 6.7.

Після перевірки застосунку на безпеку за запитом завантаження застосунку конкретним призначеним для користувача пристроєм в сховищі здійснюється другий етап схеми двохетапної компіляції і генерується інсталяційний пакет з об'єктним кодом для конкретного пристрою.

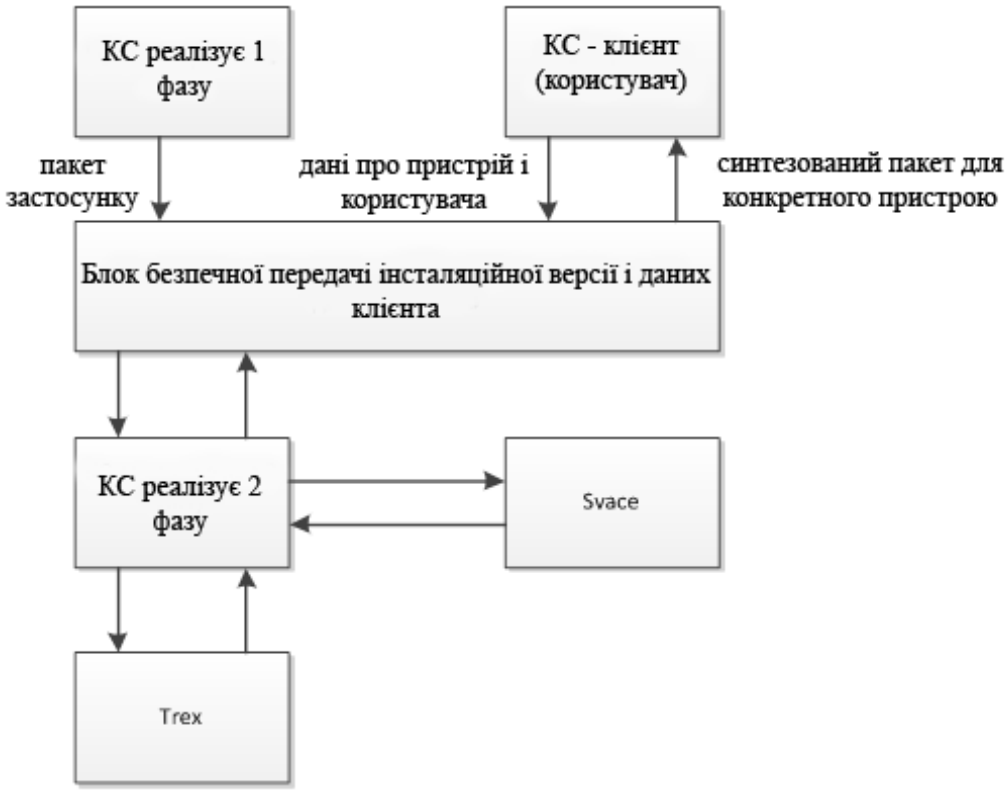


Рис. 6.7. Структурна схема системи передачі і розподілу застосунків у "хмарному сховищі"

Таким чином, розроблено метод передтестової компіляції і розподілу доступу що відрізняється від відомих врахуванням профілів користувача при синтезі застосунку, а також використанням ресурсів "хмарних сховищ" в процесі отримання інсталяційних версій. Це дозволить підвищити рівень безпеки розроблюваних застосунків.

## 6.4 Практичні рекомендації з використання методів і засобів управління безпекою застосунків

Аналіз літератури [31-38], а також проведені дослідження показали, що більшість існуючих методів і засобів управління безпекою в IT-сфері враховують можливості вже застарілої методології розроблення - "*Waterflow*".

Проте нині більшість фірм використовують гнучкі методології розроблення і управління, засновані на філософії *Agile*. Це накладає деякі особливості (обмеження) на описані в [39, 40] методи і вимагає від керівників проектів внесення деяких коригувань в процес розроблення і експлуатації.

У загальному випадку такі коригування можуть реалізуватися в наступних практичних напрямах.

1. Перегляд парадигми безпеки застосунків.

2. Розподіл команди.

3. Розширення повноважень експертів безпеки в процесі вдосконалення команди.

4. Постійний перегляд вимог, методів, засобів, алгоритмів та інших даних при управлінні безпекою.

Розглянемо детальніше кожен зі вказаних напрямів.

### 6.4.1 Вдосконалення парадигми безпеки застосунків

У ранніх парадигмах безпеки застосунків експерти звикли сприймати стан безпеки як щось особливе, окреме від усього застосунку. Проте в сучасних умовах використання гнучких методологій філософії *Agile*, безпека - це одна з властивостей якості застосунків. ІТ-фірми шукають різні підходи до управління безпекою своїх продуктів.

Наприклад, корпорація *Microsoft* представила своє бачення процесу безпечної розроблення у вигляді схеми, наведеної на рис. 6.8.

У цих умовах дуже важливим стає розуміння необхідності постійного збільшення долі автоматизованих підходів управління процесами на усіх етапах розроблення і реалізації ІТ продуктів і послуг.

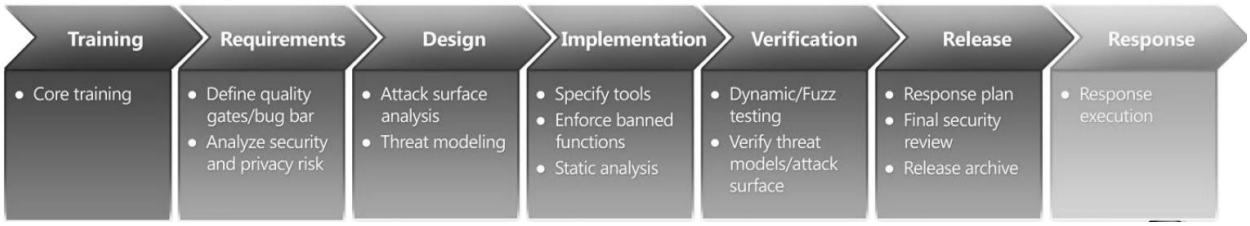


Рис. 6.8. Схема розроблення безпечного програмного забезпечення у відповідності з філософією *Microsoft*

### 6.4.2 Розподіл команд

В умовах розроблення і реалізації досить великих проектів (застосунків) існує необхідність (якщо є можливість) використання великої кількості різних команд (можливо, ще і розподілених), що виконують різні завдання. При цьому вимоги, можливості, методології розроблення і інші практики в усіх різні. Не кажучи вже про навички "безпеки". На жаль, IT-фірми дуже рідко, коли можуть собі дозволити мати експертів безпеки на кожну команду.

В той же час, вимоги максимально безпечної розроблення, без застосування методу "одна команда - один експерт безпеки" залишаються. У цих умовах доцільно скористатися способом масштабування процесу розроблення програмного забезпечення, описаним в роботах [7, 30]. Це дозволить забезпечити заданий рівень якості застосунків без залучення значних ресурсів і сил (експертів).

### 6.4.3 Розширення повноважень експертів безпеки

Однією з основних переваг гнучких методологій розроблення і управління IT-проектами є придбання і накопичення досвіду учасниками проектів.

При цьому безпека продукту - це питання розробників, їх знань і контролю, а не окремих команд і людей, тому розширення відповідальності у бік розробників, а також попередніх перевірок, може бути доцільним.

Виходячи з цього, у рамках існуючих заходів (наприклад, мітингу-ретроспективи), передбачених *Agile*, доцільне проведення додаткових тренінгів, воркшопів, мініквестів або інших заходів, основною метою яких є передача досвіду тестування експертами безпеки розробникам.

Звичайно, необхідно враховувати недолік часових ресурсів як з боку навчаних, так і з боку повчальних. Тому доцільно використати наступні правила навчання.

1. Інтерактивні модулі навчання замість нудних слайдів або відео.

2. Приклади вразливостей з реального життя і пентестів.

3. Той, що навчається, сам імітує дії того, що атакує, для розуміння суті атаки.

4. Пояснення помилок в коді для розробників.

5. Рекомендації з написання коду для розробників на тій мові програмування, яка потрібна.

Крім того, завжди є фінальне ревью, на якому команда перевіряє документацію, проводить тести безпеки, та ін. При цьому експерти безпеки стають деякою організуюче-перевіряльною ланкою, а також командою супроводу інших команд. Вони розробляють вимоги, гайди і тулзи, дають рекомендації і проводять тренінги.

Дуже важлива при цьому комунікація експертів з командою. Це дозволяє перевіряти організацію розроблення, документів і питань безпеки, організацію роботи з ключовими елементами безпеки. При цьому проводиться аналіз слабких місць команди на різних гейтах і організовується *feedback* команді.

### 6.4.4 Постійний перегляд вимог, методів, засобів, алгоритмів і інших даних при управлінні безпекою

У сучасному світі більшість IT-фірм схильна до постійної динамічної зміни, розширення і оновлення їх компонентів, а їх продукція (програмні застосунки) оновленню новими версіями. Крім того, в процесі розроблення відбуваються кадрові зміни, а з часом змінюється і політика безпеки.

Ці зміни викликають нові вразливості, а вразливості, які раніше були передбачені, можуть знову стати проблемою. Таким чином, процес управління вразливостями в IT-сфері знаходиться в постійному розвитку.

Проведені дослідження, а також аналіз літератури [8, 41-46] показали, що процес оцінки вразливості необхідно проводити з певною періодичністю.

У ряді керівних документів [47-51] ця періодичність обмежується одним разом на три роки. Проте управління вразливостями проводиться не тому, що це потрібно законодавчим актом, а тому, що це хороша практика підтримки бізнес-цілей організації.

У зв'язку з цим, графік оцінки і зменшення вразливостей, пов'язаних з безпекою, має бути досить гнучким, щоб можна було проаналізувати і оцінити зміни в IT-системі, середовищі обробки, політиці безпеки.

Відповідаючи на питання необхідності проведення тестування безпеки застосунків можна відмітити, що ряд експертів пропонує проводити тестування:

– для вже використовуваних критичних застосунків - з обраною періодичністю або при внесенні змін;

– перед запуском в експлуатацію нового бізнес-застосунку;

– при додаванні надбудов до існуючих застосунків;

– у разі інциденту інформаційної безпеки, пов'язаного з функціонуванням застосунку, і при підозрі на некоректну роботу застосунку з точки зору інформаційної безпеки.

В цьому випадку схему тестування безпеки можна представити у вигляді рис. 6.9.


Рис. 6.9. Схема тестування безпеки

Також слід зазначити, що успішна програма управління вразливостями неможлива без спільної підтримки і участі керівництва, команди і експертів безпеки.

## 6.5 Висновки до розділу

У розділі проведено дослідження ефективності розроблених моделей і методів тестування безпеки застосунків і обґрунтування практичних рекомендацій з використання методів і засобів управління безпекою.

Визначено, що використання розроблених моделей і методів дозволить від 1,05 до 1,5 разів зменшити час тестування безпеки.

При оцінці достовірності отриманих в результаті математичного моделювання даних було проведено порівняння щільності розподілу ймовірності часу тестування безпеки, відповідних меж довірчого інтервалу і оцінок його математичного сподівання. Істинне значення обраного показника потрапляє в довірчий інтервал з довірчою ймовірністю $\beta = 0{,}94$.

В якості практичного застосування в області комп'ютерної інженерії і розроблення програмних застосунків в розділі розроблено метод передтестової компіляції і розподілу доступу.

Відмінною особливістю методу є врахування профілів користувача при синтезі застосунку, а також використанням ресурсів "хмарних сховищ" в процесі отримання інсталяційних версій. Це дозволить підвищити рівень безпеки застосунків, що розробляються.

Також в розділі запропонований ряд практичних рекомендацій з управління силами і засобами безпеки ІТ-організацій в умовах використання гнучких методологій розроблення програмного забезпечення філософії *Agile*.

В цілому проведені дослідження показали, що показник безпеки ПЗ КС збільшився до 15%, що дозволяє зробити висновок про підвищення рівня захисту інформації за допомогою синтезованих моделей та методів розроблення безпечного ПЗ.

Отпубліковані наукові результати роботи автора приведені у 6 розділі [52, 53, 54, 55, 56].

**Список використаних джерел:**

[1] Panda B. Planet: Massively parallel learning of tree ensembles with mapreduce// Proceedings of the VLDB Endowment. – 2009. – Vol. 2, no. 2. – pp. 1426–1437.

[2] Panda M. Hybrid intelligent systems for detecting network intrusions/ M. Panda, A. Abraham, M. Patra : Security Comm. Networks. – 2012.– vol. 8.– pp. 2741–2749.

[3] Horia F. Pop. Costel Sarbu. A new fuzzy discriminant analysis method. MATCH Commun. Math. Comput. Chem. 69, 2013.– P.391-412.

[4] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы. Построение и анализ / Вильямс 2016, 1328 с.

[5] Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

[6] Вентцель Е.С. Теория вероятностей: Учеб. для вузов. – 6-е изд. стер. – М.: Высш. шк., 1999.– 576 с.

[7] Семенов С.Г. Методы и средства распределения доступа и защиты данных в компьютеризированных информационных управляющих системах критического применения / С.Г. Семенов. – Х.:НТУ «ХПІ», 2013. – 360 с.

[8] Zeng Y. Risk Management For Enterprise Resource Planning System Implementations in Project-Based / Y. Zeng // Firms : dis. for the degree of PHD, Maryland, 2010. – P. 210.

[9] Халіфе К. Gert-модель прогнозування параметрів функціональної безпеки технічних систем / Кассем Халіфе, С.Г. Семенов, С.Ю. Гавриленко // Зб. наукових праць. Системи обробки інформації. – Х.: ХУ ПС, 2016. – Випуск 2(139). – С.50-52.

[10] Халифе К. GERT-модель процесса безопасного тестирования программного обеспечения / Кассем Халіфе, А.Е. Горюшкіна, В.М. Зміївська // Зб. наукових праць. Системи обробки інформації. – Х.: ХУ ПС, 2016. – Випуск 3(140). – С.21-24.

[11] ДСТУ ISO/IEC 25012:2016 Інженерія систем і програмних засобів.

Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Модель якості даних (ISO/IEC 25012:2008, IDT) [Електронний ресурс] – Режим доступу: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=69135

[12] Информационная война и защита информации. Словарь основных терминов и определений [Електронний ресурс]. – Режим доступу до ресурсу: http://www.csef.ru/files/csef/articles/2176/2176.pdf

[13] Белл Л., Брантон-Сполл М., Смит Р. Безопасность разработки в Agile-проектах. ДМК Пресс, 2018. 448 с.

[14] Криспин Л., Грегори Д. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд (Signature Series). Вильямс, 2010. 464 с.

[15] Липаев В.В. Надежность и функциональная безопасность комплексов программ реального времени. Москва, 2013. 176 с.

[16] НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999р., №22. [Електронний ресурс]. – Режим доступу до ресурсу: http://do.gendocs.ru/docs/index-27508.html?page=8

[17] Нестеров С. А. Информационная безопасность.– М. Издательство Юрайт.– 2018. – 321 с.

[18] ISO 9001:1994 «Системы качества Модель для гарантии качества в проектировании, разработке, изготовлении, установки и обслуживании».

[19] Черушева Т. В. Проектирование программного обеспечения / Т. В. Черушева. – Пенза : Изд-во ПГУ, 2014. – 172 с.

[20] Чунарьова А. Методи та системи виявлення несанкціонованого

доступу в сучасних інформаційно-комунікаційних системах та мережах / А. Чунарьова, А. Чунарьов// Науково- практичний журнал «Безпека інформації». – 2012, № 1.– С. 45-49.

[21] Шибанов, А.П. Обобщенные GERT-сети для моделирования протоколов, алгоритмов и программ телекоммуникационных систем: дис. доктора техн. наук: 05.13.13 [Текст] / Шибанов Александр Петрович. – Рязань, 2003. – 307 с.

[22] Эффективное использование GNU Make Електронний ресурс]. – Режим доступу: https://www.opennet.ru/docs/RUS/gnumake/

[23] LLVM: компилятор своими руками. Введение [Електронний ресурс]. – Режим доступу до ресурсу: https://habrahabr.ru/post/277717/

[24] Гайсарян С.С., Курмангалеев Ш.Ф., Долгорукова К.Ю., Савченко В.В., Саргсян С.С. Применение метода двухфазной компиляции на основе LLVM для распространения приложений с использованием облачного хранилища [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/n/primenenie-metoda-dvuhfaznoy-kompilyatsii-na-osnove-llvm-dlya-rasprostraneniya-prilozheniy-s-ispolzovaniem-oblachnogo-hranilischa

[25] Semenov S.G., Davydov V.V., Gavrylenko Protection C.Yu. Data in computerized Governors systems. LAP Lambert Academic Publishing GmbH & Co. KG (Saarbrücken, Germany), 2014. 236 p.

[26] Semenov S., Dorokhov O., Grynov D. The concept definition of mathematical modelling of the secured information-telecommunication system with regard to conditions of the posterior uncertainty. Transport and Telecommunication.2013. Vol. 14, № 2. P. 167-174.

[27] Алгоритм Голдберга-Рао. [Електронний ресурс]. – Режим доступу: http://algolist.manual.ru/maths/graphs/maxflows/Goldberg_Rao.php

[28] Аоки М. Оптимизация стохастических систем / М. Аоки. – М.: «Наука» ФИЗМАТЛИТ, 1971. – 424 с.

[29] Гавриленко С.Ю. Методика відбору системи показників для ідентифікації стану комп'ютерної системи критичного застосування// Радіоелектронні і комп'ютерні системи/ зб. наук. пр. – Харків: НАУ «ХАІ», 2019. – №2 (90). С.127-135.

[30] Смірнов С.А. Метод антивірусного захисту даних з використанням хмарних обчислювальних технологій дис. кандидата техн. наук: 05.13.21 [Текст] / Смірнов Сергій Анатолійович. – Київ, 2017. – 128 с.

[31] Когда «Agile» (не) к месту [Електронний ресурс]. – Режим доступу до ресурсу: https://makhmetov.ru/articles/agile.html

[32] J. Highsmith Agile Software Development Ecosystems. – Boston: AddisonWesley, 2006. – 448p.

[33] Благодатских В.А. Стандартизация разработки программных средств / В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов – М: Финансы и статистика, 2003. – 288 с.

[34] Вендров А.М. Проектирование программного обеспечения экономических информационных систем / А.М. Вендров. – М.: Финансы и статистика, 2005. – 544 с.

[35] Вэйдер Майкл Томас. Инструменты бережливого производства. Мини-руководство по внедрению методик бережливого производства / Майкл Томас Вэйдер – Альпина Паблишер. 2012. – 125 с.

[36] Гусятников В.Н. Стандартизация и разработка программных систем / В.Н. Гусятников, А.И. Безруков – М: Финансы и статистика, 2010. – 288 с.

[37] Ruby Sam Agile Web Development with Rails / Sam Ruby, Dave Thomas, David Heinemeier Hansson – The Pragmatic Bookshelf, 2011. – 480 с.

[38] Seacord Robert Secure Coding in C and C++ / Robert C. Seacord – Addison-Wesley, 2013. – 600 p.

[39] Диогенес Ю.Озкайя Э. Кибербезопасность. Стратегии атак и обороны. ДМК Пресс, 2019. 326 с.

[40] Denning D. Cryptography and Data Security / D. Denning. // Addison Wesley, Reading (MA), 1983. – P. 135-185.

[41] Krishnan M. Soumya Software Development Risk Aspects and Success Frequency on Spiral and Agile Model / M. Soumya Krishnan // International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization). – 2015. –Vol.3. – №1. – pp.301-310

[42] Бриткин А. И. Риски, связанные с внедрением технологий, в проектах разработки программного обеспечения / А. Бриткин // Социально-экономические и технические системы. – 2007. – № 8 (42).

[43] Департамент QA: Ошибки управления [Електронний ресурс] – Режим доступу: http://blog.alsedi.com/departament-qa-oshibki-upravleniya/

[44] Лужецький В.А. Організаційно-правові питання безпеки інформації Концептуальна модель системи інформаційного впливу / В.А. Лужецький // Безпека інформації. Ukrainian Scientific Journal of Information Security Том 23, № 1 (2017).

[45] Макконнелл С. Совершенный код. Мастер-класс. М.: Рус. Редакция ; СПб.: Питер, 2007. 896 с.

[46] Test & Test Case Management in Jira [Електроний ресурс] – Режим доступу: http://blog.alsedi.com/test-test-case-management-in-jira-part-0/

[47] ГОСТ Р 51275-99 Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положення [Електронний ресурс]. – Режим доступу до ресурсу:

http://www.gosthelp.ru/text/GOSTR5127599Zashhitainfor.html

[48] ГОСТ Р ИСО/МЭК 15408-2-2002 Информационная технология. Методы и средства обеспечения безопасности критерии оценки безопасности информационных технологий. Часть 2 Функциональные требования безопасности [Електронний ресурс]. – Режим доступу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_15408-2-2002.htm

[49] ГОСТ Р ИСО/МЭК 13335-1-2006 Информационная технология Методы и средства обеспечения безопасности Часть 1 Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий [Електронний ресурс]. – Режим доступу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_13335-1-2006.htm

[50] ГОСТ Р ИСО/МЭК 27033-1-2011 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и концепции [Електронний ресурс]. – Режим доступу: http://protect.gost.ru/document.aspx?control=7&id=179072

[51] ГОСТ Р ИСО/МЭК 27004-2011 Информационная технология. Методы и средства обеспечения безопасности. Менеджмент информационной безопасности. [Електронний ресурс]. – Режим доступу: http://protect.gost.ru/document.aspx?control=7&id=179060

[52] Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141

[53] Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи

управління, навігації та зв’язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

[54] Коваленко О.В. Розробка методу передтестової компіляції й розподілу доступу / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницькй. 19-20 квітня 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215.

[55] Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп’ютерних систем для підвищення безпеки даних: монография / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 350 с.

[56] Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

# ВИСНОВКИ

У дисертації розв'язана науково-технічна проблема, яка є актуальною та полягає в синтезі моделей та методів розроблення безпечного ПЗ КС.

Проведений аналіз сучасних тенденцій розвитку методологій розроблення ПЗ і вимог до програмних засобів, видів математичної формалізації, а також критеріїв і показників оптимізації інформаційних процесів показав, що при впровадженні в системи критичного застосування інформаційних технологій, збільшення кількості інформації, що зберігається, обробляється і циркулює в них, а також підвищеної вразливості несанкціонованого доступу до ПЗ з боку зловмисників, використовувані нині моделі та методи розроблення ПЗ КС достатній рівень захищеності даних забезпечити не в змозі.

На основі проведеного аналізу і міжнародних та державних стандартів сформовано загальну схему характеристик і показників, що відносяться до якості ПЗ. Аналіз методологій розроблення програмного забезпечення і факторів, що впливають на безпеку, дозволив виділити протиріччя між підвищеними вимогами до безпеки ПЗ (врахуванням усіх вразливостей безпеки) і необхідністю адаптації до існуючих об'єктивно-суб'єктивних факторів, властивих сучасному світу ІТ-індустрії.

Проведені порівняльні дослідження основних підходів математичної формалізації дозволили визначити основні напрями дисертаційного дослідження і сформулювати оптимізаційне завдання синтезу моделей та методів розроблення ПЗ.

Проведені в дисертаційній роботі дослідження, результати вирішення науково-технічної проблеми і окремих наукових завдань, а також результати розрахунків і порівняльного аналізу, дали можливість отримати наступні наукові та практичні результати.

1. Удосконалено метод якісного аналізу вразливостей розроблення програмного забезпечення, що відрізняється від відомих врахуванням факторів експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки ПЗ КС і оцінкою довільного несуперечливого кінцевого набору «квантів інформації». В основу розробленого методу покладена структурна ідентифікація вразливостей розроблення ПЗ, що відрізняється від відомих побудовою оцінки вразливостей розроблення ПЗ «зверху» у вигляді множини, за наявності довільного несуперечливого кінцевого набору «квантів інформації». Це дозволило на 55% звузити сукупність множин Парето та більш точно обирати пріоритетність напрямків фінансування профілактичних заходів.

2. Удосконалено метод кількісної оцінки вразливостей розроблення ПЗ. Його відмінною особливістю є комплексне використання "Аналізу дерева відмов" і способу оцінки показника чистої приведеної вартості проекту розроблення ПЗ з урахуванням негативних факторів можливого невиявлення загроз безпеки ПЗ. Використання удосконаленого "Аналізу дерева відмов" дозволить до 20% підвищити точність кількісної оцінки вразливостей розроблення ПЗ. В той же час, використання способу оцінки показника чистої приведеної вартості проекту розроблення ПЗ дозволяє розглядати проект комплексно, з урахуванням необхідності врахування безпеки і тестування вразливості ПЗ, із залученням інструментів, які дозволяють здолати складність, невизначеність і довгостроковість проектів.

3. Удосконалено метод оптимізації розподілу ресурсів розроблення безпечного ПЗ. В основу цього методу було покладено напівмарківську модель прийняття рішень для керованого марківського процесу у безперервному часі. Відмінною особливістю запропонованого методу є використання псевдобулевих методів бівалентного програмування з нелінійною цільовою функцією і лінійними обмеженнями для визначення

оптимальної стратегії усунення експлуатаційних помилок. Це дозволило оптимізувати процес проектування стратегії управління вразливостями.

4. Розроблено математичну модель технології тестування комплексу *DOM XSS* вразливостей, яка відрізняється від відомих урахуванням специфіки комплексного аналізу різних типів *XSS* вразливості ("*stored XSS*", "*reflected XSS*" і *DOM Based XSS*), а також включенням в алгоритм процедур автоматичного аудиту *DOM Based XSS* окремо. Це надало можливість провести аналітичну оцінку часових витрат тестування вказаних вразливостей в умовах реалізації стратегії розроблення безпечного програмного забезпечення.

5. Розроблено математичну модель технології тестування вразливості до *SQL*-ін'єкцій, яка відрізняється від відомих удосконаленим способом визначення відстані між результатами ін'єкції. Використання в запропонованому методі критерію Джаро-Вінклера для порівняння результатів ін'єкції *SQL*-коду і введення порогового значення дозволить підвищити точність результатів тестування безпеки ПЗ.

6. Розроблено метод математичного моделювання технологій тестування *DOM XSS* вразливості і вразливості до *SQL*-ін'єкцій, в основу якого покладено підхід мережевого *GERT* моделювання. Це дозволить досліджувати процеси в комп'ютеризованих системах при розробці нових засобів і протоколів захисту даних, а також від 1,05 до 1,5 разів зменшити час тестування безпеки.

7. Отримано подальший розвиток імітаційної моделі технології тестування безпеки на основі положень теорії масштабування імітаційних моделей. Відмінною особливістю розробленої імітаційної моделі є адаптація вибору вхідних операторів управління і даних до підвищення вимог оперативності розроблення і реалізації моделі, виражена в реалізації процедури взаємодії з реальним браузером з використанням засобів

автоматизації браузеру і формуванні даних для атаки на декількох діалектах. Це дозволило знизити обчислювальну складність реалізованих алгоритмів до 1,5 разів.

8. Розроблено метод передтестової компіляції і розподілу доступу, що відрізняється від відомих врахуванням профілів користувача при розроблені застосунку, а також використанням ресурсів "хмарних сховищ" в процесі отримання інсталяційних версій ПЗ. Це дозволило підвищити рівень безпеки розроблених застосунків.

9. Проведено порівняльну оцінку ефективності застосування розроблених моделей та методів, а також достовірності отриманих результатів. В цілому проведені дослідження показали, що показник безпеки ПЗ КС збільшився до 15%, що дозволяє зробити висновок про підвищення рівня захисту інформації за допомогою розроблених моделей та методів розробки безпечного ПЗ.

Після проведених експериментів результат показав, що статистична величина довірчої ймовірність відхилення для видів даних що розглядаються складає $P \approx 0{,}92$, тобто значення статистичної величини від математичного сподівання «не відхилиться» більше, ніж на одиницю.

Результати дисертаційної роботи впроваджені у вигляді алгоритмів і засобів для рішення задач розроблення безпечного ПЗ КС:

– при розробці автоматизованих систем виявлення вразливостей ПЗ в Інтернет сервіс провайдері ТОВ «ІМПЕРІАЛ-НЕТ»;

– при удосконаленні гнучкої методології розроблення ПЗ у компанії-розробнику програмного забезпечення ТОВ «МІФ ПРОДЖЕКТС» (м. Кропивницький);

– при розробці систем захисту інформації ТОВ «САЙФЕР ІТ» (м. Київ);

– в учбовому процесі Центральноукраїнського національного технічного університету.

Використання результатів дисертаційної роботи підтверджене відповідними актами впровадження (Додаток А).

Наукове використання результатів, отриманих в дисертаційній роботі, можливе у рамках подальшого розвитку наукового напряму, який пов'язаний з розроблення і вдосконаленням ефективних методологій розроблення безпечного ПЗ.

# ДОДАТКИ

# ДОДАТОК А.

# ВІДОМОСТІ ЩОДО ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ

ТОВ «ІМПЕРІАЛ-НЕТ»
тел: +38 (0522) 27-60-06
факс : +38 (0522) 27-60-91
e-mail : office@imperial.net.ua
http://www.imperial.net.ua

ЗАТВЕРДЖУЮ:
Директор товариства
з обмеженою відповідальністю
«Імперіал-НЕТ»
_____ К.В. Бур'янов
«22» лютого 2019р.

АКТ

про впровадження результатів дисертаційної роботи
доцента кафедри кібербезпеки та програмного забезпечення
Центральноукраїнського національного технічного університету
Коваленко Олександра Володимировича

Комісія у складі голови – директора ТОВ «Імперіал-НЕТ» Бур'янов К.В. членів комісії – провідного розробника ТОВ «Імперіал-НЕТ» Король К.В. провідного розробника ТОВ «Імперіал-НЕТ» Дуб О.М. склала цей акт про те, що у діяльності ТОВ «Імперіал-НЕТ» отримано корисний ефект який полягає в тестуванні DOM XSS вразливостей різних типів та в тестуванні вразливостей до SQL-ін'єкцій на серверах Інтернет сервіс провайдеру Імперіал, а також можливості отримання інформації про необхідні часові характеристики тестування вищезазначених вразливостей.

ТОВ «ІМПЕРІАЛ-НЕТ»
вул. Єгорова 8
м. Кіровоград,
Україна, 25006

ПАТ КБ «ПриватБанк»
Р/Р : 26000052913562
МФО : 323583
ЄДРПОУ : 39758019

тел : +38 (0522) 27-60-06
факс : +38 (0522) 27-60-91
office@imperial.net.ua
isp@imperial.net.ua
http://www.imperial.net.ua

Реалізовано наступні результати наукових досліджень Коваленко Олександра Володимировича:

1. Впроваджено програмну реалізацію математичної моделі технології тестування комплексу DOM XSS вразливостей, яка за рахунок урахування специфіки комплексного аналізу різних типів XSS вразливості («stored XSS», «reflected XSS» і DOM Based XSS), а також включенням в алгоритм процедур автоматичного аудиту DOM Based XSS окремо, дозволяє провести аналітичну оцінку часових витрат тестування вказаних вразливостей в умовах реалізації стратегії розроблення безпечного програмного забезпечення.

2. Впроваджено програмну реалізацію математичної моделі технології тестування вразливості до SQL-ін'єкцій, яка за рахунок використання критерію Джаро-Вінклера, для порівняння результатів ін'єкції SQL-коду і введення порогового значення дозволяє підвищити точність результатів тестування безпеки програмного забезпечення.

**Голова комісії**

Директор ТОВ «Імперіал-НЕТ» К.В. Бур'янов

**Члени комісії:**

провідний розробник К.В. Король

провідний розробник О.М. Дуб

MIF PROJECTS

mifprojects.com

ЗАТВЕРДЖУЮ:
Директор товариства
з обмеженою відповідальністю
«МІФ Проджектс»
_____ А. ЛИЧУК
«28» жовтня 2019 р.

**АКТ**

про впровадження результатів дисертаційної роботи
доцента кафедри кібербезпеки та програмного забезпечення
Центральноукраїнського національного технічного університету
Коваленко Олександра Володимировича

Комісія у складі голови – Директора ТОВ «МІФ Проджектс» Личука А.М., членів комісії – провідного розробника програмного забезпечення Кобця М.О., провідного розробника програмного забезпечення Хлистуна В.В. склала цей акт про те, що у діяльності ТОВ «МІФ Проджектс» реалізовано такі результати наукових досліджень Коваленко Олександра Володимировича:

1. Метод якісного аналізу вразливостей розроблення програмного забезпечення, що відрізняється від відомих врахуванням факторів експлуатаційних вразливостей, особливо вразливості невиявлення загроз безпеки програмного забезпечення комп'ютерних систем, і оцінкою довільного несуперечливого кінцевого набору «квантів інформації», це дозволяє звузити множину важливих вразливостей і знизити можливі фінансові та іміджеві втрати організацій-розробників програмного забезпечення.

2. Метод кількісної оцінки вразливостей розроблення програмного забезпечення, що відрізняється від відомих комплексним використанням методики «Аналізу дерева відмов» і способу оцінки показника чистої приведеної вартості проекту розроблення безпечного програмного забезпечення з урахуванням негативних факторів можливого невиявлення загроз безпеки програмного забезпечення комп'ютерних систем, це дозволило підвищити точність кількісної оцінки вразливостей розроблення програмного забезпечення.

MIF PROJECTS mifprojects.com

Розроблені методи програмно реалізовані у вигляді плагіну для інструменту управління проектами для agile-команд на платформі Atlassian JIRA. Впроваджені методи дозволили проводити моніторинг та обирати пріоритетність напрямків фінансування профілактичних заходів, здолати складність, невизначеність і довгостроковість при розробці безпечного програмного забезпечення.

**Голова комісії**
Директор ТОВ «МІФ Проджектс» А. Личук

**Члени комісії:**

провідний розробник М. Кобець

провідний розробник В. Хлистун

ЗАТВЕРДЖУЮ:
Директор товариства з обмеженою
відповідальністю «Сайфер ІТ»
_____ В.Ю. Ковтун
« 11 » грудня 2019 р.

**АКТ**

про впровадження результатів дисертаційної роботи
доцента кафедри кібербезпеки та програмного забезпечення
Центральноукраїнського національного технічного університету
Коваленко Олександра Володимировича

Комісія у складі голови – Директора ктн Ковтун В.Ю., членів комісії – керівника проектів ктн Боровікова О.М., провідного розробника Бойко С.Т. склала цей акт про те, що у діяльності ТОВ «Сайфер ІТ» реалізовано наступний результат наукових досліджень Коваленко Олександра Володимировича:

Отримало подальший розвиток імітаційної моделі технології тестування безпеки web-застосувань та web-сервісів, що реалізують REST API на основі положень теорії масштабування імітаційних моделей. Відмінною особливістю розробленої імітаційної моделі є адаптація вибору вхідних операторів управління і даних до підвищення вимог оперативності розроблення і реалізації моделі, виражена в реалізації процедури взаємодії з реальним web-браузером, з використанням засобів автоматизації web-браузеру і формуванні даних для атаки на декількох діалектах, що дозволило понизити обчислювальну складність алгоритмів тестування, що реалізовуються.

У результаті виконання наукових досліджень Коваленко Олександра Володимировича, запропонована імітаційна модель технології тестування безпеки web-застосувань системи дистанційного обслуговування клієнтів банку ELPay та web-сервісів «Шифр-CaaS», «Шифр-Arch» , що реалізують REST API на основі положень теорії масштабування імітаційних моделей, показала що для коректного моделювання даного процесу досить використати слабкіше поняття транзитивної залежності за управлінням.

Це істотно знижує обчислювальну складність алгоритму масштабування, що на практиці значно зменшує час тестування релізів розгалужених web-застосувань у контексті безпечної розробки програмних продуктів ТОВ «Сайфер ІТ».

**Голова комісії**
Директор ТОВ «Сайфер ІТ» ктн, В.Ю. Ковтун

**Члени комісії:**
Керівник проектів ктн, О.М. Боровіков

провідний розробник С.Т. Бойко

ЗАТВЕРДЖУЮ:
Проректор з наукової роботи
Центральноукраїнського національного
технічного університету
_____ О. Левченко
«__» жовтня 2019 р.

**АКТ**

реалізації результатів наукових досліджень дисертаційної роботи доцента кафедри кібербезпеки та програмного забезпечення Коваленко Олександра Володимировича

Комісія у складі голови – доцента кафедри кібербезпеки та програмного забезпечення, кандидата технічних наук, Босько В.В., членів комісії – доцента кафедри кібербезпеки та програмного забезпечення, кандидата технічних наук, Доренського О.П., доцента кафедри кібербезпеки та програмного забезпечення, кандидата фізико-математичних наук, Якименко Н.М. склала цей акт про те, що при розробці лекційних, практичних та лабораторних занять з навчальних дисциплін «Комп'ютерні мережі», «Системи керування проектами» у навчальному процесі Центральноукраїнського національного технічного університету були використані наступні результати наукових досліджень Коваленко Олександра Володимировича:

1. Метод передтестової компіляції і розподілу доступу, який за рахунок врахування профілів користувача при розроблені застосунку, а також використанням ресурсів «хмарних сховищ» в процесі отримання інсталяційних версій дозволяє підвищити рівень безпеки застосунків, що розробляються.

2. Метод оптимізації розподілу ресурсів розроблення ПЗ на основі напівмарківської моделі прийняття рішень для керованого марківського процесу у безперервному часі. Відмінною особливістю запропонованого методу є використання псевдобулевих методів біваленттного програмування з нелінійною цільовою функцією і лінійними обмеженнями для визначення оптимальної стратегії усунення експлуатаційних помилок, це дозволяє оптимізувати процес проектування стратегії розподілу ресурсів розроблення ПЗ. Розроблено програмне забезпечення, яке реалізує даний метод.

Застосування результатів дисертаційних досліджень Коваленко Олександра Володимировича дозволило підвищити рівень засвоєння навчального матеріалу з дисциплін «Комп'ютерні мережі» та «Системи керування проектами» за рахунок більш поглибленого вивчення сучасних моделей та методів розроблення безпечного програмного забезпечення комп'ютерних систем.

**Голова комісії**
доцент кафедри кібербезпеки та
програмного забезпечення,
кандидат технічних наук В.В. Босько

**Члени комісії:**
доцент кафедри кібербезпеки та
програмного забезпечення,
кандидат технічних наук О.П. Доренський

доцент кафедри кібербезпеки та
програмного забезпечення,
кандидат фізико-математичних наук Н.М. Якименко

# ДОДАТОК Б.
# АНАЛІТИЧНІ РОЗВ'ЯЗКИ МЕТОДУ ОПТИМІЗАЦІЇ РОЗПОДІЛУ РЕСУРСІВ РОЗРОБЛЕННЯ БЕЗПЕЧНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

```
-->  /* p - ймовірність переходу системи зі стану i в стан j;, f – значення потоку події для функції розподілу часу перебування системи в стані i при прийнятті рішення r_i за умови, що наступний перехід відбудеться в стан j. Тут формується система рівнянь, яка задана формулами (3.1). Параметри системи відповідають структурі наданій на рис. 3.2 та таблиці 3.1, 3.2, 3.3, 3.4. */;
-->  Q[i,i]: (1-sum( sum( p[i,j,r]·f[i,j,r], r, 0, 1), j, 0, 5))/s+
     sum(sum(p[i,j,r]·f[i,j,r]·Q[j,i], r, 0, 1), j, 0, 5);
-->  /* Тут і далі «%o» позначає відповідь системи математичної обробки "wxMaxima" */;
```

(%o1) $(-f_{i,5,1}p_{i,5,1}-f_{i,5,0}p_{i,5,0}-f_{i,4,1}p_{i,4,1}-f_{i,4,0}p_{i,4,0}-f_{i,3,1}p_{i,3,1}-f_{i,3,0}p_{i,3,0}-f_{i,2,1}p_{i,2,1}-f_{i,2,0}p_{i,2,0}-f_{i,1,1}p_{i,1,1}-f_{i,1,0}p_{i,1,0}-f_{i,0,1}p_{i,0,1}-f_{i,0,0}p_{i,0,0}+1)/s+Q_{5,i}f_{i,5,1}p_{i,5,1}+Q_{5,i}f_{i,5,0}p_{i,5,0}+Q_{4,i}f_{i,4,1}p_{i,4,1}+Q_{4,i}f_{i,4,0}p_{i,4,0}+Q_{3,i}f_{i,3,1}p_{i,3,1}+Q_{3,i}f_{i,3,0}p_{i,3,0}+Q_{2,i}f_{i,2,1}p_{i,2,1}+Q_{2,i}f_{i,2,0}p_{i,2,0}+Q_{1,i}f_{i,1,1}p_{i,1,1}+Q_{1,i}f_{i,1,0}p_{i,1,0}+Q_{0,i}f_{i,0,1}p_{i,0,1}+Q_{0,i}f_{i,0,0}p_{i,0,0}$

```
-->  Q[i,j]: sum(sum(p[i,k,r]·f[i,k,r]·Q[k,j], r, 0, 1), k, 0, 5);
```

(%o2) $Q_{5,j}f_{i,5,1}p_{i,5,1}+Q_{5,j}f_{i,5,0}p_{i,5,0}+Q_{4,j}f_{i,4,1}p_{i,4,1}+Q_{4,j}f_{i,4,0}p_{i,4,0}+Q_{3,j}f_{i,3,1}p_{i,3,1}+Q_{3,j}f_{i,3,0}p_{i,3,0}+Q_{2,j}f_{i,2,1}p_{i,2,1}+Q_{2,j}f_{i,2,0}p_{i,2,0}+Q_{1,j}f_{i,1,1}p_{i,1,1}+Q_{1,j}f_{i,1,0}p_{i,1,0}+Q_{0,j}f_{i,0,1}p_{i,0,1}+Q_{0,j}f_{i,0,0}p_{i,0,0}$

```
-->  /* Далі система записана у вигляді перетворення Лапласа, як показано у виразі системою (3.26) */;
-->  eqf0: [f[0,0,0]=0, f[0,0,1]=0, f[0,1,0]=0.3·0.2/(s+0.3), f[0,1,1]=0,
     f[0,2,0]=0.15·0.2/(s+0.15), f[0,2,1]=0, f[0,3,0]=0.1·0.2/(s+0.1),
     f[0,3,1]=0, f[0,4,0]=0.08/(s+0.4),f[0,4,1]=0,
     f[0,5,0]=0.15·0.2/(s+0.15),
     f[0,5,1]=0];
     eqf1: [f[1,0,0]=1/(s+1), f[1,0,1]=2/(s+2), f[1,1,0]=0, f[1,1,1]=0,
     f[1,2,0]=0, f[1,2,1]=0, f[1,3,0]=0, f[1,3,1]=0, f[1,4,0]=0, f[1,4,1]=0,
     f[1,5,0]=0, f[1,5,1]=0];
     eqf2: [f[2,0,0]=4/(s+4), f[2,0,1]=10/(s+10), f[2,1,0]=0,
     f[2,1,1]=0, f[2,2,0]=0, f[2,2,1]=0, f[2,3,0]=0, f[2,3,1]=0, f[2,4,0]=0,
     f[2,4,1]=0,
     f[2,5,0]=0, f[2,5,1]=0];
     eqf3: [f[3,0,0]=20/(s+20), f[3,0,1]=1/(s+1), f[3,1,0]=0, f[3,1,1]=0,
     f[3,2,0]=0, f[3,2,1]=0, f[3,3,0]=0, f[3,3,1]=0, f[3,4,0]=0, f[3,4,1]=0,
```

```
      f[3,5,0]=0, f[3,5,1]=0];
      eqf4: [f[4,0,0]=20/(s+20/7)/7, f[4,0,1]=4/(s+4), f[4,1,0]=0, f[4,1,1]=0,
      f[4,2,0]=0, f[4,2,1]=0, f[4,3,0]=0, f[4,3,1]=0, f[4,4,0]=0, f[4,4,1]=0,
      f[4,5,0]=0, f[4,5,1]=0];
      eqf5: [f[5,0,0]=1/(s+1), f[5,0,1]=2/(s+2), f[5,1,0]=0, f[5,1,1]=0,
      f[5,2,0]=0, f[5,2,1]=0, f[5,3,0]=0, f[5,3,1]=0, f[5,4,0]=0, f[5,4,1]=0,
      f[5,5,0]=0, f[5,5,1]=0];
```

--> /* Середовище програмування відображає систему в образі Лапласа рівнянь у наступному вигляді: */;

(eqf0) $[f_{0,0,0}=0, f_{0,0,1}=0, f_{0,1,0}=\frac{0.06}{s+0.3}, f_{0,1,1}=0, f_{0,2,0}=\frac{0.03}{s+0.15}, f_{0,2,1}=0, f_{0,3,0}=\frac{0.02}{s+0.1}, f_{0,3,1}=0, f_{0,4,0}=\frac{0.08}{s+0.4}, f_{0,4,1}=0, f_{0,5,0}=\frac{0.03}{s+0.15}, f_{0,5,1}=0]$

(eqf1) $[f_{1,0,0}=\frac{1}{s+1}, f_{1,0,1}=\frac{2}{s+2}, f_{1,1,0}=0, f_{1,1,1}=0, f_{1,2,0}=0, f_{1,2,1}=0, f_{1,3,0}=0, f_{1,3,1}=0, f_{1,4,0}=0, f_{1,4,1}=0, f_{1,5,0}=0, f_{1,5,1}=0]$

(eqf2) $[f_{2,0,0}=\frac{4}{s+4}, f_{2,0,1}=\frac{10}{s+10}, f_{2,1,0}=0, f_{2,1,1}=0, f_{2,2,0}=0, f_{2,2,1}=0, f_{2,3,0}=0, f_{2,3,1}=0, f_{2,4,0}=0, f_{2,4,1}=0, f_{2,5,0}=0, f_{2,5,1}=0]$

(eqf3) $[f_{3,0,0}=\frac{20}{s+20}, f_{3,0,1}=\frac{1}{s+1}, f_{3,1,0}=0, f_{3,1,1}=0, f_{3,2,0}=0, f_{3,2,1}=0, f_{3,3,0}=0, f_{3,3,1}=0, f_{3,4,0}=0, f_{3,4,1}=0, f_{3,5,0}=0, f_{3,5,1}=0]$

(eqf4) $[f_{4,0,0}=\frac{20}{7\left(s+\frac{20}{7}\right)}, f_{4,0,1}=\frac{4}{s+4}, f_{4,1,0}=0, f_{4,1,1}=0, f_{4,2,0}=0, f_{4,2,1}=0, f_{4,3,0}=0, f_{4,3,1}=0, f_{4,4,0}=0, f_{4,4,1}=0, f_{4,5,0}=0, f_{4,5,1}=0]$

(eqf5) $[f_{5,0,0}=\frac{1}{s+1}, f_{5,0,1}=\frac{2}{s+2}, f_{5,1,0}=0, f_{5,1,1}=0, f_{5,2,0}=0, f_{5,2,1}=0, f_{5,3,0}=0, f_{5,3,1}=0, f_{5,4,0}=0, f_{5,4,1}=0, f_{5,5,0}=0, f_{5,5,1}=0]$

--> /* Задаємо ймовірності згідно таблиці схемі 3.2 з позначенням змінними шуканих ймовірностей ($p_{X,0,0}$): */;

```
-->   eqp0: [p[0,0,0]=0, p[0,0,1]=0, p[0,1,0]=1, p[0,1,1]=0, p[0,2,0]=1,
      p[0,2,1]=0, p[0,3,0]=1, p[0,3,1]=0, p[0,4,0]=1, p[0,4,1]=0, p[0,5,0]=1,
      p[0,5,1]=0];
      eqp1: [p[1,0,0]=p100, p[1,0,1]=p101, p[1,1,0]=0, p[1,1,1]=0, p[1,2,0]=0,
      p[1,2,1]=0, p[1,3,0]=0, p[1,3,1]=0, p[1,4,0]=0, p[1,4,1]=0, p[1,5,0]=0,
      p[1,5,1]=0];
      eqp2: [p[2,0,0]=p200, p[2,0,1]=p201, p[2,1,0]=0, p[2,1,1]=0, p[2,2,0]=0,
      p[2,2,1]=0, p[2,3,0]=0, p[2,3,1]=0, p[2,4,0]=0, p[2,4,1]=0, p[2,5,0]=0,
      p[2,5,1]=0];
      eqp3: [p[3,0,0]=p300, p[3,0,1]=p301, p[3,1,0]=0, p[3,1,1]=0, p[3,2,0]=0,
      p[3,2,1]=0, p[3,3,0]=0, p[3,3,1]=0, p[3,4,0]=0, p[3,4,1]=0, p[3,5,0]=0,
      p[3,5,1]=0];
      eqp4: [p[4,0,0]=p400, p[4,0,1]=p401, p[4,1,0]=0, p[4,1,1]=0, p[4,2,0]=0,
      p[4,2,1]=0, p[4,3,0]=0, p[4,3,1]=0, p[4,4,0]=0, p[4,4,1]=0, p[4,5,0]=0,
      p[4,5,1]=0];
      eqp5: [p[5,0,0]=p500, p[5,0,1]=p501, p[5,1,0]=0, p[5,1,1]=0, p[5,2,0]=0,
      p[5,2,1]=0, p[5,3,0]=0, p[5,3,1]=0, p[5,4,0]=0, p[5,4,1]=0, p[5,5,0]=0,
      p[5,5,1]=0];
```

(eqp0) $[p_{0,0,0}=0, p_{0,0,1}=0, p_{0,1,0}=1, p_{0,1,1}=0, p_{0,2,0}=1, p_{0,2,1}=0, p_{0,3,0}=1, p_{0,3,1}=0, p_{0,4,0}=1, p_{0,4,1}=0, p_{0,5,0}=1, p_{0,5,1}=0]$

(eqp1) $[p_{1,0,0}=p100, p_{1,0,1}=p101, p_{1,1,0}=0, p_{1,1,1}=0, p_{1,2,0}=0, p_{1,2,1}=0, p_{1,3,0}=0, p_{1,3,1}=0, p_{1,4,0}=0, p_{1,4,1}=0, p_{1,5,0}=0, p_{1,5,1}=0]$

(eqp2) $[p_{2,0,0}=p200, p_{2,0,1}=p201, p_{2,1,0}=0, p_{2,1,1}=0, p_{2,2,0}=0, p_{2,2,1}=0, p_{2,3,0}=0, p_{2,3,1}=0, p_{2,4,0}=0, p_{2,4,1}=0, p_{2,5,0}=0, p_{2,5,1}=0]$

(eqp3) $[p_{3,0,0}=p300, p_{3,0,1}=p301, p_{3,1,0}=0, p_{3,1,1}=0, p_{3,2,0}=0, p_{3,2,1}=0, p_{3,3,0}=0, p_{3,3,1}=0, p_{3,4,0}=0, p_{3,4,1}=0, p_{3,5,0}=0, p_{3,5,1}=0]$

(eqp4) $[p_{4,0,0}=p400, p_{4,0,1}=p401, p_{4,1,0}=0, p_{4,1,1}=0, p_{4,2,0}=0, p_{4,2,1}=0, p_{4,3,0}=0, p_{4,3,1}=0, p_{4,4,0}=0, p_{4,4,1}=0, p_{4,5,0}=0, p_{4,5,1}=0]$

(eqp5) $[p_{5,0,0}=p500, p_{5,0,1}=p501, p_{5,1,0}=0, p_{5,1,1}=0, p_{5,2,0}=0, p_{5,2,1}=0, p_{5,3,0}=0, p_{5,3,1}=0, p_{5,4,0}=0, p_{5,4,1}=0, p_{5,5,0}=0, p_{5,5,1}=0]$

```
-->   q00: Q[i,i], i=0;
      q10: Q[i,j], i=1, j=0;
      q20: Q[i,j], i=2, j=0;
      q30: Q[i,j], i=3, j=0;
      q40: Q[i,j], i=4, j=0;
      q50: Q[i,j], i=5, j=0;
```

```
-->   /* Змушуємо математичне середовище застосувати індекси для отримання
записів конкретних компонентів системи рівнянь: */;
```

(q00) $(-f_{0,5,1}p_{0,5,1}-f_{0,5,0}p_{0,5,0}-f_{0,4,1}p_{0,4,1}-f_{0,4,0}p_{0,4,0}-f_{0,3,1}p_{0,3,1}-f_{0,3,0}p_{0,3,0}-f_{0,2,1}p_{0,2,1}-f_{0,2,0}p_{0,2,0}-f_{0,1,1}p_{0,1,1}-f_{0,1,0}p_{0,1,0}-f_{0,0,1}p_{0,0,1}-f_{0,0,0}p_{0,0,0}+1)/s+f_{0,5,1}p_{0,5,1}Q_{5,0}+f_{0,5,0}p_{0,5,0}Q_{5,0}+f_{0,4,1}p_{0,4,1}Q_{4,0}+f_{0,4,0}p_{0,4,0}Q_{4,0}+f_{0,3,1}p_{0,3,1}Q_{3,0}+f_{0,3,0}p_{0,3,0}Q_{3,0}+f_{0,2,1}p_{0,2,1}Q_{2,0}+f_{0,2,0}p_{0,2,0}Q_{2,0}+f_{0,1,1}p_{0,1,1}Q_{1,0}+f_{0,1,0}p_{0,1,0}Q_{1,0}+Q_{0,0}f_{0,0,1}p_{0,0,1}+Q_{0,0}f_{0,0,0}p_{0,0,0}$

(q10) $f_{1,5,1}p_{1,5,1}Q_{5,0}+f_{1,5,0}p_{1,5,0}Q_{5,0}+f_{1,4,1}p_{1,4,1}Q_{4,0}+f_{1,4,0}p_{1,4,0}Q_{4,0}+f_{1,3,1}p_{1,3,1}Q_{3,0}+f_{1,3,0}p_{1,3,0}Q_{3,0}+f_{1,2,1}p_{1,2,1}Q_{2,0}+f_{1,2,0}p_{1,2,0}Q_{2,0}+Q_{1,0}f_{1,1,1}p_{1,1,1}+Q_{1,0}f_{1,1,0}p_{1,1,0}+Q_{0,0}f_{1,0,1}p_{1,0,1}+Q_{0,0}f_{1,0,0}p_{1,0,0}$

(q20) $f_{2,5,1}p_{2,5,1}Q_{5,0}+f_{2,5,0}p_{2,5,0}Q_{5,0}+f_{2,4,1}p_{2,4,1}Q_{4,0}+f_{2,4,0}p_{2,4,0}Q_{4,0}+f_{2,3,1}p_{2,3,1}Q_{3,0}+f_{2,3,0}p_{2,3,0}Q_{3,0}+Q_{2,0}f_{2,2,1}p_{2,2,1}+Q_{2,0}f_{2,2,0}p_{2,2,0}+Q_{1,0}f_{2,1,1}p_{2,1,1}+Q_{1,0}f_{2,1,0}p_{2,1,0}+Q_{0,0}f_{2,0,1}p_{2,0,1}+Q_{0,0}f_{2,0,0}p_{2,0,0}$

(q30) $f_{3,5,1}p_{3,5,1}Q_{5,0}+f_{3,5,0}p_{3,5,0}Q_{5,0}+f_{3,4,1}p_{3,4,1}Q_{4,0}+f_{3,4,0}p_{3,4,0}Q_{4,0}+Q_{3,0}f_{3,3,1}p_{3,3,1}+Q_{3,0}f_{3,3,0}p_{3,3,0}+Q_{2,0}f_{3,2,1}p_{3,2,1}+Q_{2,0}f_{3,2,0}p_{3,2,0}+Q_{1,0}f_{3,1,1}p_{3,1,1}+Q_{1,0}f_{3,1,0}p_{3,1,0}+Q_{0,0}f_{3,0,1}p_{3,0,1}+Q_{0,0}f_{3,0,0}p_{3,0,0}$

(q40) $f_{4,5,1}p_{4,5,1}Q_{5,0}+f_{4,5,0}p_{4,5,0}Q_{5,0}+Q_{4,0}f_{4,4,1}p_{4,4,1}+Q_{4,0}f_{4,4,0}p_{4,4,0}+Q_{3,0}f_{4,3,1}p_{4,3,1}+Q_{3,0}f_{4,3,0}p_{4,3,0}+Q_{2,0}f_{4,2,1}p_{4,2,1}+Q_{2,0}f_{4,2,0}p_{4,2,0}+Q_{1,0}f_{4,1,1}p_{4,1,1}+Q_{1,0}f_{4,1,0}p_{4,1,0}+Q_{0,0}f_{4,0,1}p_{4,0,1}+Q_{0,0}f_{4,0,0}p_{4,0,0}$

(q50) $Q_{5,0}f_{5,5,1}p_{5,5,1}+Q_{5,0}f_{5,5,0}p_{5,5,0}+Q_{4,0}f_{5,4,1}p_{5,4,1}+Q_{4,0}f_{5,4,0}p_{5,4,0}+Q_{3,0}f_{5,3,1}p_{5,3,1}+Q_{3,0}f_{5,3,0}p_{5,3,0}+Q_{2,0}f_{5,2,1}p_{5,2,1}+Q_{2,0}f_{5,2,0}p_{5,2,0}+Q_{1,0}f_{5,1,1}p_{5,1,1}+Q_{1,0}f_{5,1,0}p_{5,1,0}+Q_{0,0}f_{5,0,1}p_{5,0,1}+Q_{0,0}f_{5,0,0}p_{5,0,0}$

```
-->   /* Пов'язуємо групи виразів у остаточну систему рівнянь: */;
```

```
-->   qf00: Q[0,0] = q00, eqf0, eqp0;
      qf10: Q[1,0] = q10, eqf0, eqf1, eqp0, eqp1;
      qf20: Q[2,0] = q20, eqf0, eqf2, eqp0, eqp2;
      qf30: Q[3,0] = q30, eqf0, eqf3, eqp0, eqp3;
      qf40: Q[4,0] = q40, eqf0, eqf4, eqp0, eqp4;
      qf50: Q[5,0] = q50, eqf0, eqf5, eqp0, eqp5;
```

```
-->   /* Система рівнянь приймає наступний вигляд: */;
```

(qf00) $$Q_{0,0}=\frac{0.08\,Q_{4,0}}{s+0.4}+\frac{-\frac{0.08}{s+0.4}-\frac{0.06}{s+0.3}-\frac{0.06}{s+0.15}-\frac{0.02}{s+0.1}+1}{s}+\frac{0.06\,Q_{1,0}}{s+0.3}+\frac{0.03\,Q_{5,0}}{s+0.15}+\frac{0.03\,Q_{2,0}}{s+0.15}+\frac{0.02\,Q_{3,0}}{s+0.1}$$

(qf10) $$Q_{1,0}=\frac{2\,Q_{0,0}\,p101}{s+2}+\frac{Q_{0,0}\,p100}{s+1}$$

(qf20) $$Q_{2,0}=\frac{10\,Q_{0,0}\,p201}{s+10}+\frac{4\,Q_{0,0}\,p200}{s+4}$$

(qf30) $$Q_{3,0}=\frac{20\,Q_{0,0}\,p300}{s+20}+\frac{Q_{0,0}\,p301}{s+1}$$

(qf40) $$Q_{4,0}=\frac{4\,Q_{0,0}\,p401}{s+4}+\frac{20\,Q_{0,0}\,p400}{7\left(s+\frac{20}{7}\right)}$$

(qf50) $$Q_{5,0}=\frac{2\,Q_{0,0}\,p501}{s+2}+\frac{Q_{0,0}\,p500}{s+1}$$

--> /* Тут $Q_{i,j}$ – є шуканими образами Лапласа для вираження ймовірностей еволюції ймовірності знаходження системи у відповідних i, j станах (розділ 3, п.1). Остаточно формуємо систему функціональних рівнянь, звівши подібні члени: */;

```
-->   rt0: solve( [qf00, qf10, qf20, qf30, qf40, qf50], [Q[0,0],
        Q[1,0], Q[2,0], Q[3,0],
        Q[4,0], Q[5,0]]);
-->   rt0[1][1];
-->   q01: Q[i,j], i=0, j=1$
      q11: Q[i,i], i=1$
      q21: Q[i,j], i=2, j=1$
      q31: Q[i,j], i=3, j=1$
      q41: Q[i,j], i=4, j=1$
      q51: Q[i,j], i=5, j=1$
      qf01: Q[0,1] = q01, eqf0, eqf1, eqp0, eqp1;
      qf11: Q[1,1] = q11, eqf1, eqp1;
      qf21: Q[2,1] = q21, eqf1, eqf2, eqp1, eqp2;
      qf31: Q[3,1] = q31, eqf1, eqf3, eqp1, eqp3;
      qf41: Q[4,1] = q41, eqf1, eqf4, eqp1, eqp4;
      qf51: Q[5,1] = q51, eqf1, eqf5, eqp1, eqp5;
```

(qf01) $$Q_{0,1}=\frac{0.08\,Q_{4,1}}{s+0.4}+\frac{0.06\,Q_{1,1}}{s+0.3}+\frac{0.03\,Q_{5,1}}{s+0.15}+\frac{0.03\,Q_{2,1}}{s+0.15}+\frac{0.02\,Q_{3,1}}{s+0.1}$$

(qf11) $$Q_{1,1}=\frac{-\frac{2\,p101}{s+2}-\frac{p100}{s+1}+1}{s}+\frac{2\,Q_{0,1}\,p101}{s+2}+\frac{Q_{0,1}\,p100}{s+1}$$

(qf21) $$Q_{2,1}=\frac{10\,Q_{0,1}\,p201}{s+10}+\frac{4\,Q_{0,1}\,p200}{s+4}$$

(qf31) $$Q_{3,1}=\frac{20\,Q_{0,1}\,p300}{s+20}+\frac{Q_{0,1}\,p301}{s+1}$$

(qf41) $$Q_{4,1}=\frac{4\,Q_{0,1}\,p401}{s+4}+\frac{20\,Q_{0,1}\,p400}{7\left(s+\frac{20}{7}\right)}$$

(qf51) $$Q_{5,1}=\frac{2\,Q_{0,1}\,p501}{s+2}+\frac{Q_{0,1}\,p500}{s+1}$$

```
-->   /* Далі робота продовжується з наступними компонентами системи: */;

-->   rt1: solve( [qf01, qf11, qf21, qf31, qf41, qf51],
       [Q[0,1], Q[1,1], Q[2,1],
        Q[3,1], Q[4,1], Q[5,1]])$
-->   rt1[1][1];

-->   q02: Q[i,j], i=0, j=2$
      q12: Q[i,j], i=1, j=2$
      q22: Q[i,i], i=2$
      q32: Q[i,j], i=3, j=2$
      q42: Q[i,j], i=4, j=2$
      q52: Q[i,j], i=5, j=2$
      qf02: Q[0,2] = q02, eqf2, eqf0, eqp2, eqp0$
      qf12: Q[1,2] = q12, eqf2, eqf1, eqp2, eqp1$
      qf22: Q[2,2] = q22, eqf2, eqp2$
      qf32: Q[3,2] = q32, eqf2, eqf3, eqp2, eqp3$
      qf42: Q[4,2] = q42, eqf2, eqf4, eqp2, eqp4$
      qf52: Q[5,2] = q52, eqf2, eqf5, eqp2, eqp5$
      rt2: solve( [qf02, qf12, qf22, qf32, qf42, qf52], [Q[0,2], Q[1,2],
      Q[2,2], Q[3,2], Q[4,2], Q[5,2]])$
      rt2[1][1];

-->   q03: Q[i,j], i=0, j=3$
      q13: Q[i,j], i=1, j=3$
      q23: Q[i,j], i=2, j=3$
      q33: Q[i,i], i=3$
      q43: Q[i,j], i=4, j=3$
      q53: Q[i,j], i=5, j=3$
      qf03: Q[0,3] = q03, eqf3, eqf0, eqp3, eqp0$
      qf13: Q[1,3] = q13, eqf3, eqf1, eqp3, eqp1$
      qf23: Q[2,3] = q23, eqf3, eqf2, eqp3, eqp2$
      qf33: Q[3,3] = q33, eqf3, eqp3$
      qf43: Q[4,3] = q43, eqf3, eqf4, eqp3, eqp4$
      qf53: Q[5,3] = q53, eqf3, eqf5, eqp3, eqp5$
      rt3: solve( [qf03, qf13, qf23, qf33, qf43, qf53], [Q[0,3], Q[1,3],
      Q[2,3], Q[3,3], Q[4,3], Q[5,3]])$
      rt3[1][1];
```

```
-->   q04: Q[i,j], i=0, j=4$
      q14: Q[i,j], i=1, j=4$
      q24: Q[i,j], i=2, j=4$
      q34: Q[i,j], i=3, j=4$
      q44: Q[i,i], i=4$
      q54: Q[i,j], i=5, j=4$
      qf04: Q[0,4] = q04, eqf4, eqf0, eqp4, eqp0$
      qf14: Q[1,4] = q14, eqf4, eqf1, eqp4, eqp1$
      qf24: Q[2,4] = q24, eqf4, eqf2, eqp4, eqp2$
      qf34: Q[3,4] = q34, eqf4, eqf3, eqp4, eqp3$
      qf44: Q[4,4] = q44, eqf4, eqp4$
      qf54: Q[5,4] = q54, eqf4, eqf5, eqp4, eqp5$
      rt4: solve( [qf04, qf14, qf24, qf34, qf44, qf54], [Q[0,4], Q[1,4],
      Q[2,4], Q[3,4], Q[4,4], Q[5,4]])$
      rt4[1][1];

-->   q05: Q[i,j], i=0, j=5$
      q15: Q[i,j], i=1, j=5$
      q25: Q[i,j], i=2, j=5$
      q35: Q[i,j], i=3, j=5$
      q45: Q[i,j], i=4, j=5$
      q55: Q[i,i], i=5$
      qf05: Q[0,5] = q05, eqf5, eqf0, eqp5, eqp0$
      qf15: Q[1,5] = q15, eqf5, eqf1, eqp5, eqp1$
      qf25: Q[2,5] = q25, eqf5, eqf2, eqp5, eqp2$
      qf35: Q[3,5] = q35, eqf5, eqf3, eqp5, eqp3$
      qf45: Q[4,5] = q45, eqf5, eqf4, eqp5, eqp4$
      qf55: Q[5,5] = q55, eqf5, eqp5$
      rt5: solve( [qf05, qf15, qf25, qf35, qf45, qf55], [Q[0,5], Q[1,5],
      Q[2,5], Q[3,5], Q[4,5], Q[5,5]])$
      rt5[1][1];

-->   test00: [p100=0.0, p101=1.0, p200=1.0, p201=0.0, p300=0.0, p301=1.0,
      p400=1.0, p401=0.0, p500=0.0, p501=1.0]$
      r0: rt0[1][1], test00;
      r1: rt1[1][1], test00;
      r2: rt2[1][1], test00;
      r3: rt3[1][1], test00;
      r4: rt4[1][1], test00;
      r5: rt5[1][1], test00;

-->   /* Проводимо спрощення запису результатів для наочності: */;
```

```
-->   partfrac( last(r0), s );
      partfrac( last(r1), s );
      partfrac( last(r2), s );
      partfrac( last(r3), s );
      partfrac( last(r4), s );
      partfrac( last(r5), s );
```

--> /* В результаті для кожного зі станів $Q_{0,(0,1,..,5)}$ маємо наступні образи Лапласа: */;

$$(\%o120)\quad \frac{27300000\,s^6+221654000\,s^5+587615600\,s^4+589473100\,s^3+211500325\,s^2+29590650\,s+2119200}{953\,(350000\,s^7+3782500\,s^6+15094000\,s^5+27698025\,s^4+23967780\,s^3+9427076\,s^2+1601232\,s+91488)}+\frac{875}{953\,s}$$

$$(\%o121)\quad \frac{15}{953\,s}-\frac{5250000\,s^6+36724500\,s^5+56156550\,s^4-54992370\,s^3-125275353\,s^2-50289810\,s-4480032}{953\,(350000\,s^7+3782500\,s^6+15094000\,s^5+27698025\,s^4+23967780\,s^3+9427076\,s^2+1601232\,s+91488)}$$

$$(\%o122)\quad \frac{15}{1906\,s}-\frac{5250000\,s^6+36724500\,s^5+93180600\,s^4+106326705\,s^3+53392134\,s^2+8313972\,s-248712}{1906\,(350000\,s^7+3782500\,s^6+15094000\,s^5+27698025\,s^4+23967780\,s^3+9427076\,s^2+1601232\,s+91488)}$$

$$(\%o123)\quad \frac{30}{953\,s}-\frac{10500000\,s^6+106804000\,s^5+388063650\,s^4+611488675\,s^3+410570172\,s^2+114401932\,s+10709856}{953\,(350000\,s^7+3782500\,s^6+15094000\,s^5+27698025\,s^4+23967780\,s^3+9427076\,s^2+1601232\,s+91488)}$$

$$(\%o124)\quad \frac{21}{1906\,s}-\frac{7350000\,s^6+26064500\,s^5-85954400\,s^4-375763395\,s^3-368416216\,s^2-105775376\,s-8161272}{1906\,(350000\,s^7+3782500\,s^6+15094000\,s^5+27698025\,s^4+23967780\,s^3+9427076\,s^2+1601232\,s+91488)}$$

$$(\%o125)\quad \frac{15}{953\,s}-\frac{5250000\,s^6+46731000\,s^5+139782300\,s^4+167695140\,s^3+83717547\,s^2+14209230\,s+94368}{953\,(350000\,s^7+3782500\,s^6+15094000\,s^5+27698025\,s^4+23967780\,s^3+9427076\,s^2+1601232\,s+91488)}$$

--> /* Складові виду A/s є образами константних складових, що, в разі стійкості системи, означає асимтотичне наближення значення ймовірності до коефіцієнту A спостерігати систему у зазначеному стані. Перевірити правильність викладок та обчислення можна склавши коефіцієнти, які дають 1 (бо система гарантовано знаходиться в одному зі станів). */;

```
-->   875·2+30+15+60+21+30;
```

(%o127) 1906

--> /* Сума чисельників з образів констант рівні знаменнику – дріб рівний одиниці. Перевірка показала на правдоподібність отриманого результату. Також потрібно пересвідчитися в стійкості отриманої системи. Для цього достатньо перевірити,що всі дійсні частини коренів знаменника додаткових дробів є від'ємними: */;

```
-->   allroots(350000·s^7+3782500·s^6+15094000·s^5+27698025·
      s^4+23967780·s^3+9427076·s^2+1601232·s+91488);
```

(%o126) [$s=-0.1155158298861653$, $s=-0.2469790740250694$, $s=-0.3644826095508095$, $s=-1.014683267916086$,

$s=-2.082102568819642, s=-2.949845629044096, s=-4.03353387790099]$

```
-->   /* Змінимо набір прийнятих рішень, як ймовірності переходів, і повторимо
розв'язання по вже введеним системам: */;

-->   test01: [p100=0.0, p101=1.0, p200=0.0, p201=1.0, p300=0.0, p301=1.0,
      p400=1.0, p401=0.0, p500=0.0, p501=1.0]$
      r0: rt0[1][1], test01$
      r1: rt1[1][1], test01$
      r2: rt2[1][1], test01$
      r3: rt3[1][1], test01$
      r4: rt4[1][1], test01$
      r5: rt5[1][1], test01$
      partfrac( last(r0), s );
      partfrac( last(r1), s );
      partfrac( last(r2), s );
      partfrac( last(r3), s );
      partfrac( last(r4), s );
      partfrac( last(r5), s );
```

$$(\%o165)\quad \frac{7350000\,s^6+102665500\,s^5+340022200\,s^4+372481775\,s^3+134500925\,s^2+18696600\,s+1528500}{271\,(350000\,s^7+5882500\,s^6+29326000\,s^5+60914625\,s^4+56316360\,s^3+22875770\,s^2+3949944\,s+227640)}+\frac{250}{271\,s}$$

$$(\%o166)\quad \frac{30}{1897\,s}-\frac{10500000\,s^6+136638000\,s^5+301858950\,s^4-186350055\,s^3-585184827\,s^2-246663912\,s-22297020}{1897\,(350000\,s^7+5882500\,s^6+29326000\,s^5+60914625\,s^4+56316360\,s^3+22875770\,s^2+3949944\,s+227640)}$$

$$(\%o167)\quad \frac{6}{1897\,s}-\frac{2100000\,s^6+15376500\,s^5+43355700\,s^4+57803835\,s^3+33219093\,s^2+4790904\,s-452940}{1897\,(350000\,s^7+5882500\,s^6+29326000\,s^5+60914625\,s^4+56316360\,s^3+22875770\,s^2+3949944\,s+227640)}$$

$$(\%o168)\quad \frac{60}{1897\,s}-\frac{21000000\,s^6+339671000\,s^5+1550984850\,s^4+2763334925\,s^3+1962821778\,s^2+561821516\,s+53291160}{1897\,(350000\,s^7+5882500\,s^6+29326000\,s^5+60914625\,s^4+56316360\,s^3+22875770\,s^2+3949944\,s+227640)}$$

$$(\%o169)\quad \frac{3}{271\,s}-\frac{1050000\,s^6+10059500\,s^5-14839400\,s^4-115009245\,s^3-125271826\,s^2-37138028\,s-2901240}{271\,(350000\,s^7+5882500\,s^6+29326000\,s^5+60914625\,s^4+56316360\,s^3+22875770\,s^2+3949944\,s+227640)}$$

$$(\%o170)\quad \frac{30}{1897\,s}-\frac{10500000\,s^6+156556500\,s^5+587831700\,s^4+777648435\,s^3+407553213\,s^2+70893888\,s+466980}{1897\,(350000\,s^7+5882500\,s^6+29326000\,s^5+60914625\,s^4+56316360\,s^3+22875770\,s^2+3949944\,s+227640)}$$

```
-->   250/271+30/1897+6/1897+60/1897+3/271+30/1897, numer;
```

(%o171) 1.0

```
-->   /* Тепер маємо інші ймовірності смостерігати систему у відповідних
станах, а також повна ймовірність равна одиниці. Тепер математичне середовище
здатне отримувати рішення для різного набору заданих рішень: */;

-->   test02: [p100=0.0, p101=1.0, p200=0.1, p201=0.9, p300=0.0, p301=1.0,
      p400=1.0, p401=0.0, p500=0.0, p501=1.0]$
      r0: rt0[1][1], test02$
      r1: rt1[1][1], test02$
      r2: rt2[1][1], test02$
```

```
r3: rt3[1][1], test02$
r4: rt4[1][1], test02$
r5: rt5[1][1], test02$
partfrac( last(r0), s );
partfrac( last(r1), s );
partfrac( last(r2), s );
partfrac( last(r3), s );
partfrac( last(r4), s );
partfrac( last(r5), s );
```

(%o181) $$\frac{2588250000\,s^7+46547172500\,s^6+264507079000\,s^5+610416022125\,s^4+572757141125\,s^3+196806533500\,s^2+27019450500\,s+2137830000}{18979\,(1750000\,s^8+36412500\,s^7+264311500\,s^6+891302825\,s^5+1500360885\,s^4+1241187883\,s^3+477474604\,s^2+80175276\,s+4554960)}+\frac{17500}{18979\,s}$$

(%o182) $$\frac{300}{18979\,s}-\frac{525000000\,s^7+8930955000\,s^6+42412508250\,s^5+51014589825\,s^4-66628317645\,s^3-129463578300\,s^2-50476651380\,s-4461466800}{18979\,(1750000\,s^8+36412500\,s^7+264311500\,s^6+891302825\,s^5+1500360885\,s^4+1241187883\,s^3+477474604\,s^2+80175276\,s+4554960)}$$

(%o183) $$\frac{69}{18979\,s}-\frac{120750000\,s^7+1516065000\,s^6+7020904500\,s^5+15595862100\,s^4+17482845405\,s^3+8906203980\,s^2+1256435460\,s-93964800}{18979\,(1750000\,s^8+36412500\,s^7+264311500\,s^6+891302825\,s^5+1500360885\,s^4+1241187883\,s^3+477474604\,s^2+80175276\,s+4554960)}$$

(%o184) $$\frac{600}{18979\,s}-\frac{1050000000\,s^7+21183235000\,s^6+145496134750\,s^5+448448596375\,s^4+650981835730\,s^3+420790924780\,s^2+115073266840\,s+10663394400}{18979\,(1750000\,s^8+36412500\,s^7+264311500\,s^6+891302825\,s^5+1500360885\,s^4+1241187883\,s^3+477474604\,s^2+80175276\,s+4554960)}$$

(%o185) $$\frac{210}{18979\,s}-\frac{367500000\,s^7+4989565000\,s^6+8874012000\,s^5-61102093150\,s^4-205002524720\,s^3-188490667260\,s^2-53037381040\,s-4063626000}{18979\,(1750000\,s^8+36412500\,s^7+264311500\,s^6+891302825\,s^5+1500360885\,s^4+1241187883\,s^3+477474604\,s^2+80175276\,s+4554960)}$$

(%o186) $$\frac{300}{18979\,s}-\frac{525000000\,s^7+9927352500\,s^6+60703519500\,s^5+156459066975\,s^4+175923302355\,s^3+85063650300\,s^2+14203780620\,s+93493200}{18979\,(1750000\,s^8+36412500\,s^7+264311500\,s^6+891302825\,s^5+1500360885\,s^4+1241187883\,s^3+477474604\,s^2+80175276\,s+4554960)}$$

# ДОДАТОК В.

## Список публікацій здобувача за темою дисертації та відомості про апробацію результатів дисертації

**Наукові праці, в яких опубліковані основні наукові результати дисертації**

[1] Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe **(Scopus)**

[2] Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». *CEUR Workshop Proceedings* Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe **(Scopus)**

[3] Kovalenko Oleksandr Qualitative risk analysis of software development / Oleksandr Kovalenko, Jamil Al-Azzeh, Oleksii Smirnov, Anna Kovalenko, Serhii Smirnov // Asian Journal of Information Technology. – Volume 17 Issue 3. – Medwell Journals. – 2018. – P. 218-230. ISSN: 1682-3915. URL: http://medwelljournals.com/abstract/?doi=ajit.2018.218.230 Doi: ajit.2018. 218.230

[4] Kovalenko Oleksandr, The mathematical model of the testing technology for Dom XSS vulnerabilities / O. Kovalenko, O. Smirnov, A.Kovalenko, S. Smirnov, V. Vialkova // Scientific & practical cyber security journal (SPCSJ) Volume 2 Issue 1, P. 22-28. Georgia. Tbilisi. Scientific Cyber

Security Association (SCSA), 2018 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/04-3-o.kovalenko-a.kovalenko-o.smirnov-s.smirnov-v.vialkova.pdf

[5] Коваленко А.В. Технология тестирования DOM XSS уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific & practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

[6] Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп'ютерних систем для підвищення безпеки даних: монография / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 350 с.

[7] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Информационные технологии в управлении, образовании, науке и промышленности: монографія / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

[8] Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

[9] Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

[10] Коваленко А.В. Задачи распознавания ситуаций в ERP системах/ А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

[11] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 5(142). – Х.: ХУПС – 2016. – С. 153-157.

[12] Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". – Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

[13] Коваленко А.В. Метод качественного анализа рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

[14] Коваленко А.В. Метод количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

[15] Коваленко А.В. Использование псевдобулевых методов бивалентного программирования для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

[16] Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко // Системи управління, навігації

та зв’язку. – Випуск 2 (38). – Полтава: ПолтНТУ. – 2016. – С. 93-100.

[17] Коваленко А.В. Технология тестирования уязвимости к SQL инъекциям / А.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 5 (45). – Полтава: ПолтНТУ. – 2017. – С. 66-71.

[18] Коваленко А.В. Масштабирование имитационной модели технологии тестирования безопасности / А.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 6 (46). – Полтава: ПолтНТУ. – 2017. – С. 181-184.

[19] Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

[20] Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розроблення програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

[21] Коваленко О.В. Управління ризиками розроблення програмного забезпечення за умови обмеженості коштів виділених на усунення помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький: ЦНТУ. - 2018. – С. 128-140.

[22] Коваленко О.В. GERT-мережевий синтез технології тестування на вразливість WEB-додатків/ О.В. Коваленко // Захист інформації. – Випуск 20(2) – К.: НАУ. – 2018. – С. 89-94.

[23] Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

[24] Коваленко О.В. Оцінка ефективності технології тестування

безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141.

[25] Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

[26] Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

[27] Коваленко О.В. Аналіз та дослідження інформаційних технологій розроблення програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

[28] Коваленко О.В. Удосконалений метод управління ризиками розроблення програмного забезпечення на основі напівмарковської моделі прийняття рішень / О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2 (3 ). – Харків. – 2018. – С. 48-41.

[29] Коваленко О.В. Математичні моделі технології тестування DOM XSS вразливості та вразливості до SQL ін'єкцій / О.В. Коваленко // Вісник Черкаського державного технологічного університету. Серія: Технічні науки №4, 2018. – С. 29-36.

[30] Коваленко О.В. Математична модель технології тестування вразливості до SQL ін'єкцій / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (58). – Полтава: ПолтНТУ. – 2019. – С. 43-47.

[31] Коваленко О.В. Математична модель технології тестування комплексу DOM XSS вразливостей для аналітичної оцінки часових витрат / О.В. Коваленко // Центральноукраїнський науковий вісник. Технічні науки.

№ 2(33). с. 173-180, 2019.

**Наукові праці, які засвідчують апробацію матеріалів дисертації**

[1] Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe **(Scopus)**.

[2] Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». *CEUR Workshop Proceedings* Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe **(Scopus)**.

[3] Коваленко А.В. Исследование источников и причин риска разработки программного обеспечения, этапов и работ, при выполнении которых возникает риск / А.А. Смирнов, А.В. Коваленко // Збірник тез VII всеукраїнської науково-практичної конференції "Інформатика та системні науки (ІСН-2016)". м. Полтава. 10-12 березня 2016 р. – Полтава.: ПУЕТ – 2016. – С. 264-266.

[4] Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

[5] Коваленко А.В. Методика структурной идентификации рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко //

Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 24-25 березня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 71-72.

[6] Коваленко А.В. Методы качественного анализа рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016). м. Харків. 30 березня – 1 квітня 2016 р. – Харків: НТУ «ХПІ». – 2016. – С. 6-7.

[7] Коваленко А.В. Структурная идентификация рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 15-16 квітня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 175-182.

[8] Коваленко А.В. Исследование разработанной методики структурной идентификации рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез VIII міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 28-29 квітня 2016 р. – Харків: ХНЕУ. – 2016. – С. 49.

[9] Коваленко А.В. Исследование дерева рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез ІІІ міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХНМІ ДВНЗ «УБС». – 2016. – С. 174-178.

[10] Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Сборник тезисов XII международной конференции "Стратегия качества в

промышленности и образовании". г. Варна. Болгария. 30 мая – 02 июня 2016 г – Варна. ТУВ. – 2016. – С. 585-589.

[11] Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Кібербезпека в Україні: правові та організаційні питання». м. Одеса, 21 жовтня 2016 р. – Одеса : ОДУВС, 2016. – С.146-148.

[12] Коваленко А.В. Метод управления рисками разработки программного обеспечения с использованием псевдобулевых методов бивалентного программирования / А.А. Смирнов, А.В. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Актуальні задачі та досягнення у галузі кібербезпеки». м. Кропивницький, 23-25 листопада 2016 року – Кропивницький: ЦНТУ, 2016. – С. 162.

[13] Коваленко А.В. Псевдобулевые методы бивалентного программирования для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко, С.А. Смирнов // Збірник наукових праць III міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 22-25 лютого 2017 р. – Київ: Європейський університет. – 2017. – С. 158-162.

[14] Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко // Збірник тез II науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 23-24 березня 2017 р. – Київ: КНУ ім. Тараса Шевченко – 2017. – С. 203-205.

[15] Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Conferenta internationala (editia a XIII-a).

«Securitea informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

[16] Коваленко А.В. Алгоритм анализа DOM XSS уязвимости при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез дев'ятнадцятого міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кропивницький 7-8 квітня 2017 р. – Кропивницький: ГЛА НАУ. – 2017. – С. 125-127.

[17] Коваленко А.В. Алгоритм анализа уязвимости SQL Injection для управления рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез другої міжнародної науково-технічної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2017). м. Харків. 10-12 квітня 2017 р. – Харків: НТУ «ХПІ». – 2017. – С. 27.

[18] Коваленко А.В. Метод управления рисками разработки программного обеспечения на основе алгоритмов анализа уязвимостей / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 20-22 квітня 2017 р. – Кіровоград: КНТУ. – 2017. – С. 92.

[19] Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.А. Смирнов, А.В. Коваленко, А.С. Коваленко // Збірник тез IX міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. – 2017. – С. 61.

[20] Kovalenko O.V. Method of testing the DOM XSS vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii //

International Conference «information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. – 2017. – P. 7.

[21] Коваленко О.В. Метод тестування DOM XSS уразливості / О.В. Коваленко, О.А. Смірнов, А.С. Коваленко, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

[22] Коваленко О.В. GERT-модель технології тестування DOM XSS уразливості / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць IV міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 21-24 лютого 2018 р. – Київ: Європейський університет. – 2018. – С. 65-70.

[23] Коваленко О.В. Технології тестування уразливостей Web-застосунків з використанням GERT-моделі / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". м. Кривий Ріг. 21-23 березня 2018 р. – Кривий Ріг.: ДВНЗ КНУ – 2018. – С. 227-230.

[24] Коваленко А.В. Тестирование уязвимости Web-приложений к атаке вида межсайтовый скриптинг / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов, // Збірник тез «Securitea informationala 2018». Conferenta internationala (editia a XIV-a). Chisinau. Moldova. 20-21 martie 2018. – Chisinau: ADSEM. – 2018. – P. 54-56.

[25] Коваленко А.В. Комплекс математических моделей технологии

тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов, // Збірник тез X міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

[26] Коваленко О.В. Розробка методу передтестової компіляції й розподілу доступу / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницькй. 19-20 квітня 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215.

[27] Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL-ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

[28] Коваленко О.В. Аналіз основних підходів математичного моделювання та методологій для забезпечення максимальних показників безпеки програмного забезпечення / О.В. Коваленко, А.С. Коваленко // Збірник наукових праць всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених «Комп’ютерна інженерія і кібербезпека : досягнення та інновації, м. Кропивницькй. 27-29 листопада 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 74.

**Апробацію результатів дисертації проведено на:**

– II міжнародна науково-практична конференція «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». 2016 р.. Київ – очна участь;

– Conferenta internationala (editia a XII-a). «Securitea informationala

2015-2016». 2016. Chisinau. Moldova – очна участь;

– VII всеукраїнська науково-практична конференція "Інформатика та системні науки (ІСН-2016)". 2016 р.. Полтава – очна участь;

– Науково-практична конференція “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. 2016 р. Київ – очна участь;

– Міжнародна науково-практична конференція «Інформаційна безпека та комп'ютерні технології» (IS&CT). 2016 р. Кіровоград – очна участь;

– Перша міжнародна науково-практична конференція «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016). 2016 р. Харків – очна участь;

– XVIII міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування». 2016 р. Кіровоград – очна участь;

– VIII міжнародна науково-практична конференція “Проблеми і перспективи розвитку ІТ-індустрії”. 2016 р. Харків – очна участь;

– III міжнародна науково-практична конференція «Інформаційна та економічна безпека» (INFECO-2016)». 2016 р. Харків – очна участь;

– XII международная конференция "Стратегия качества в промышленности и образовании". 2016 г. Варна, Болгария – заочна участь;

– Всеукраїнська науково-практична конференція «Кібербезпека в Україні: правові та організаційні питання». 2016 р. Одеса – очна участь;

– Всеукраїнська науково-практична конференція «Актуальні задачі та досягнення у галузі кібербезпеки». 2016 р. Кропивницький – очна участь;

– III міжнародна науково-практична конференція «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». 2017 р. Київ – очна участь;

– II науково-практична конференція “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. 2017 р. Київ – очна участь;

Conferenta internationala (editia a XIII-a). «Securitea informationala

2017». 2017. Chisinau. Republic of Moldova – очна участь;

– Дев’ятнадцятий міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування». 2017 р. Кропивницький – очна участь;

– Друга міжнародна науково-технічна конференція «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2017). 2017 р. Харків – очна участь;

– Міжнародна науково-практична конференція «Інформаційна безпека та комп'ютерні технології» (IS&CT). 2017 р. Кіровоград – очна участь;

– IX міжнародна науково-практична конференція “Проблеми і перспективи розвитку IT-індустрії”. 2017 р. Харків – очна участь;

– International Conference «information technologies, systems and networks ITSN-2017». 2017. Chisinau, Republic of Moldova – заочна участь;

– Всеукраїнська науково-практична інтернет-конференція «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». 2017 р. Кропивницький – очна участь;

– IV міжнародна науково-практична конференція «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». 2018 р. Київ – очна участь;

– Всеукраїнська науково-практична конференція "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". 2018 р. Кривий Ріг – очна участь;

Conferenta internationala (editia a XIV-a). «Securitea informationala 2018». 2018. Chisinau. Moldova – заочна участь;

– X міжнародна науково-практична конференція “Проблеми і перспективи розвитку IT-індустрії”. 2018 р. Харків – очна участь;

– III міжнародна науково-практична конференція “Інформаційна

безпека та комп’ютерні технології”, 2018 р. Кропивницький – очна участь;

– Международная конференция "Стратегия качества в промышленности и образовании" 2018 г. Варна, Болгария – заочна участь;

– Всеукраїнська науково-практична конференція здобувачів вищої освіти та молодих учених “Комп’ютерна інженерія і кібербезпека: досягнення та інновації” 2018 р. Кропивницький – очна участь.