Вісник ЛДУБЖД
Bulletin of Lviv State University of Life Safety
https://journal.ldubgd.edu.ua/index.php/Visnuk
ISSN 2078-4643 (print), ISSN 2708-1389 (online)

UDC 004.413
DOI: 10.32447/20784643.27.2023.03

Ю. С. Кордунова[1], М. Фелтіновскі[2], О. В. Придатко[1], О. О. Смотр[1]
[1]Львівський державний університет безпеки життєдіяльності, м. Львів, Україна
[2]Головна школа пожежної служби, м. Варшава, Польща
ORCID: https://orcid.org/0000-0003-0151-8285 – Ю. С. Кордунова
https://orcid.org/0000-0001-5614-8387 – М. Фелтіновскі
https://orcid.org/0000-0002-0719-9118 – О. В. Придатко
https://orcid.org/0000-0003-2767-5019 – О. О. Смотр
kordunovayulia@gmail.com

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСУ РОЗРОБКИ СПЕЦІАЛІЗОВАНИХ ПРОГРАМНИХ СИСТЕМ БЕЗПЕКО-ОРІЄНТОВАНОГО СПРЯМУВАННЯ

Проблема. Процес створення спеціалізованих програмних систем безпеко-орієнтованого спрямування потребує значних зусиль та ресурсів. Відомі на сьогодні підходи до розробки програмних систем використовують лише теоретичні відомості про етапи життєвого циклу програмного забезпечення, не занурюючись у можливу їх оптимізацію та автоматизацію.

Мета. Метою роботи є побудова математичної моделі, яка описує процес розробки спеціалізованих програмних систем безпеко-орієнтованого спрямування, що ґрунтується на використанні понятійного апарату теорії множин, а також визначення основних взаємозв'язків окремих етапів розробки означених програмних систем для автоматизації менеджменту їх життєвого циклу.

Методи дослідження. У статті розглядається модель процесу розробки програмного забезпечення, представлення якої базується на використанні математичного апарату теорії множин із додатковою візуалізацією шляхом геометричного моделювання за допомогою кругів Ейлера.

Основні результати дослідження. Зважаючи на актуальну науково-прикладну задачу з автоматизації окремих процесів управління життєвим циклом програмних систем, в роботі було проведено детальних аналіз та моделювання усіх етапів розробки програмного продукту із використанням понятійного апарату теорії множин. Представлена в статті математична модель дає підстави сформувати повну та чітку уяву про обсяги робіт, виконання яких лежить в основі успішної реалізації програмних систем, у тому числі безпеко-орієнтованого спрямування. Отримана модель дає змогу охарактеризувати окремі етапи розробки продукту та аналітично описати життєвий цикл програмного забезпечення, що є передумовою для пошуку шляхів автоматизації окремих процесів підтримки прийняття управлінських рішень.

Висновки та конкретні пропозиції авторів. Проведені теоретичні дослідження дали можливість отримати математичну модель життєвого циклу програмних систем з використанням понятійного апарату теорії множин. Ця модель дозволяє оцінити обсяг робіт на кожному етапі розробки програмного продукту, встановити залежності та взаємозв'язки між ними, а також сформувати фундаментальні принципи автоматизації процедури підтримки прийняття управлінських рішень на різних етапах створення програмних систем, зокрема безпеко-орієнтованого спрямування.

Ключові слова: спеціалізовані програмні системи, життєвий цикл програмного забезпечення, математичний апарат теорії множин, математична модель процесу розробки програмних систем.

Yu. S. Kordunova[1], M. Feltynowski[2], O. V. Prydatko[1], O. O. Smotr[1]
[1]Lviv State University of Life Safety, Lviv, Ukraine
[2]Szkoła Główna Służby Pożarniczej

MATHEMATICAL MODELING OF SPECIALIZED SAFETY-ORIENTED SOFTWARE SYSTEMS DEVELOPMENT PROCESS

Introduction. The process of creating specialised safety-oriented software systems requires considerable effort and resources. The currently known approaches to software system development use only theoretical information about the stages of the software life cycle, without diving into their possible optimisation and automation.

Purpose. The purpose of the paper is to build a mathematical model describing the process of developing specialised safety-oriented software systems based on the conceptual apparatus of set theory. As well as to determine the main

interrelationships of individual stages of this software systems development as well as the definition of the main interrelationships of individual stages of the specialised software systems development for the automation of their life cycle management.

Methods. In this article, we consider a model of the software development process, the representation of which is based on the use of the mathematical apparatus of set theory with additional visualisation by geometric modelling using Euler diagrams.

Results. Given the urgent scientific and applied task of automating certain processes of managing the software systems life cycle, the paper conducts a detailed analysis and modelling of all stages of software product development using the conceptual framework of set theory. The mathematical model presented in the article gives grounds to form a complete and clear idea of the scope of work, the implementation of which is the basis for the successful implementation of software systems, including safety-oriented ones. The resulting model allows for characterising individual stages of product development and analytically describing the software life cycle, which is a prerequisite for finding ways to automate certain processes of management decision support.

Conclusion. The theoretical studies carried out allowed us to obtain a mathematical model of the life cycle of software systems using the conceptual apparatus of set theory. This model makes it possible to estimate the amount of work at each stage of software product development, to establish dependencies and relationships between them, and to formulate fundamental principles for automating the procedure for supporting management decision-making at various stages of software systems development, in particular, safety-oriented ones.

Keywords: specialised software systems, software life cycle, mathematical apparatus of set theory, a mathematical model of the software system development process.

Вступ. Глобальні світові та загальнодержавні процеси цифровізації стимулюють до розроблення низки програмних систем, зокрема соціального характеру. Ризики, пов'язані з повномасштабним вторгненням та військовими діями на території України, значно активізували процеси розробки програмних систем безпеко-орієнтованого спрямування. Основне призначення подібних систем – допомога населенню у ризикових ситуаціях, надання консультаційної допомоги, своєчасне інформування (оповіщення) про загрозу та низка іншого функціоналу, орієнтованого на підвищення рівня безпеки громад та громадян. Поряд із реалізацією подібних соціальних проектів підприємствами ІТ-індустрії, завдання з розробки безпеко-орієнтованих сервісів покладаються і на окремі структурні підрозділи оперативних формувань, зокрема Державної служби України з надзвичайних ситуацій (ДСНС України). В рамках виконання ряду науково-дослідних робіт на замовлення ДСНС України, розробкою подібних сервісів займаються на базі Львівського державного університету безпеки життєдіяльності. Очевидно, що процес створення спеціалізованих програмних систем безпеко-орієнтованого спрямування є складним та ресурсоємним, особливо в частині формування проєктної команди, підбору фахівців та управління їх беклогом на усіх етапах розробки, а також імплементації програмного продукту. Зважаючи на те, що розробка спеціалізованого програмного забезпечення потребує залучення значних матеріальних та людських ресурсів у найкоротші терміни, існує потреба в оптимізації процесів управління життєвим циклом безпеко-орієнтованих систем та автоматизації їх окремих етапів.

Станом на сьогодні існує велика кількість публікацій, присвячених процесу розробки програмних систем. Зокрема, у роботі [1] запропоновано залучати когнітивні технології на рівні експертних систем оцінки часу розробки, прийняття рішень щодо вибору архітектури, побудови програмних застосунків тощо. В науковій праці [2] автори розглянули процес розробки програмного забезпечення, який складається із чотирьох фаз, де архітектура програмного забезпечення є найголовнішою, оскільки забезпечує абстрактне представлення загальної структури програмної системи. У роботі [3] було розроблено модель системи управління якістю процесу розробки програмного забезпечення на основі процесного підходу PDCA. У дослідженні [4] автори розробили методи якісного аналізу та кількісної оцінки ризиків розробки програмного забезпечення. У статті [5] було використано теорію обмежень у контексті процесу розробки програмного забезпечення. В дослідженнях [6-9], [14-17] основна увага приділена вибору методології для управління процесами розробки програмного забезпечення безпеко-орієнтованого спрямування. В окремих роботах [11-13] описано процес створення сервісів, орієнтованих на забезпечення якості підготовки рятувальників, командами у динамічному оточенні, проте в згаданих роботах не зазначено, які саме моделі управління життєвим циклом програмних систем були застосовані при їх реалізації.

Проведений аналіз предметної області вказує на відсутність у згаданих дослідженнях результатів, орієнтованих на автоматизацію окремих процесів менеджменту життєвого циклу програмних систем, натомість у дослідженнях використовувались лише теоретичні відомості про означені процеси. Зважаючи на недосліджену раніше проблематику, з метою вивчення

можливості автоматизації процесів управління життєвим циклом програмних систем, зокрема безпеко-орієнтованого програмування, постає науково-прикладна задача моделювання етапів розробки спеціалізованих програмних систем із використанням понятійного апарату теорії множин. Отримана математична модель створює передумови для комплексно аналізу черговості усіх етапів, визначення їх взаємозв'язків та взаємозалежностей, а також дає можливість сформувати чітку уяву про сутність кожного етапу і системну поведінку у динамічних умовах.

Методи досліджень. У статті розглядається модель процесу розробки програмних систем, представлення якої базується на використанні математичного апарату теорії множин із додатковою візуалізацією шляхом геометричного моделювання за допомогою кругів Ейлера. Зазначений підхід дає змогу охарактеризувати окремі етапи процесу розробки програмних систем та математично описати життєвий цикл програмного забезпечення, шляхом застосування математичних операцій об'єднання та перетину множин проєктних робіт.

Метою роботи є побудова математичної моделі, яка описує процес розробки спеціалізованих програмних систем безпеко-орієнтованого спрямування, що ґрунтується на використанні понятійного апарату теорії множин, а також визначення основних взаємозв'язків окремих етапів розробки означених програмних систем для автоматизації менеджменту їх життєвого циклу. Отримана математична модель надає змістовні підстави для проведення досліджень щодо оптимізації ресурсної складової проєктів, шляхом забезпечення якості продукту за умови зниження часу та матеріальних витрат на розробку, а також підвищення ефективності процесів управління життєвим циклом безпеко-орієнтованих програмних систем.

Результати досліджень. Для вирішення поставленої мети існує потреба у побудові геометричної моделі життєвого циклу спеціалізованих програмних систем. На рисунку 1 зображений універсум <U>, елементами якого є множини, що імітують етапи розробки програмного забезпечення.

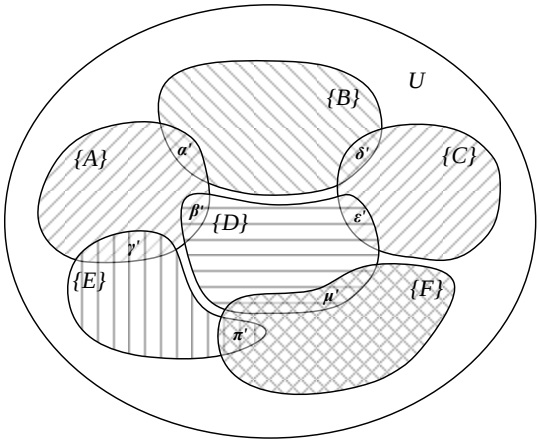

Рисунок 1 – Геометрична модель життєвого циклу спеціалізованих програмних систем у вигляді універсуму <U>

Вміст універсуму можна описати таким чином:

$$U = \{A, B, C, D, E, F\}, \quad (1)$$

де: *{A}* – множина обсягу робіт з аналізу предметної області і формулювання вимог;

{B} – множина обсягу робіт з проєктування архітектури програми та UX-дизайну;

{C} – множина обсягу робіт з реалізації програмної системи в кодах;

{D} – множина обсягу робіт з тестування програмної системи;

{E} – множина обсягу робіт з впровадження програмної системи;

{F} – множина обсягу робіт з супроводу програмної системи під час експлуатації.

В універсумі <U> між множинами існують як операції перетину, так і об'єднання, зокрема:

$$A\cap B; A\cap D; A\cap E; A\cup C; A\cup F; B\cap C; B\cup D; B\cup F; C\cap D; C\cup A; C\cup E; C\cup F; D\cap F; D\cup E; E\cap F; E\cup B; E\cup C. \quad (2)$$

В результаті перетину множин в межах універсуму <U> отримано такі результати:

$$A\cap B = \{\alpha' : \alpha' \in A \; i \; \alpha' \in B\}, \quad (3)$$

де: α' – визначені вимоги до програмного забезпечення на основі аналізу предметної області і формулювання системних вимог;

$$A\cap D = \{\beta' : \beta' \in A \; i \; \beta' \in D\}, \quad (4)$$

де: β' – вимоги до тестування програмного забезпечення, які визначаються на етапі аналізу предметної області і формулювання системних вимог (перевірка відповідності цим вимогам);

$$A \cap E = \{\gamma' : \gamma' \in A \; i \; \gamma' \in E\}, \quad (5)$$

де : γ' – вимоги до впровадження програми, які визначаються на етапі аналізу предметної області і формулювання системних вимог (до прикладу вибір платформи для розгортання продукту, визначення етапів розгортання продукту на стороні замовника, визначення етапів навчання користувачів тощо);

$$B \cap C = \{\delta' : \delta' \in B \; i \; \delta' \in C\}, \quad (6)$$

де: δ' – програмний код, що реалізує функціонал програмної системи згідно із специфікацією та поданою архітектурою;

$$C \cap D = \{\varepsilon' : \varepsilon' \in C \; i \; \varepsilon' \in D\}, \quad (7)$$

де: ε' – результати тестування програмного коду на відповідність специфікації та результати опрацювання журналу помилок (допомагає виявляти проблеми в роботі програмної системи та їх природу);

$$D \cap F = \{\mu' : \mu' \in D \; i \; \mu' \in F\}, \quad (8)$$

де: μ' – технічні звіти за результатами тестування, які містять інформацію про виявлені проблеми, шляхи їх вирішення та рекомендації щодо тестування і виправлення недоліків під час експлуатації програмної системи;

$$E \cap F = \{\pi' : \pi' \in E \; i \; \pi' \in F\}, \quad (9)$$

де: π' – вимоги до процесів підтримки програмного забезпечення.

Для глибшого аналізу процесу розробки програмного забезпечення розглянемо кожну із множин універсуму <U>. На рисунку 2 зображена модель множини А, яка відтворює етап аналізу предметноїобласті і формулювання системних вимог (постановки завдання) до програмного продукту.

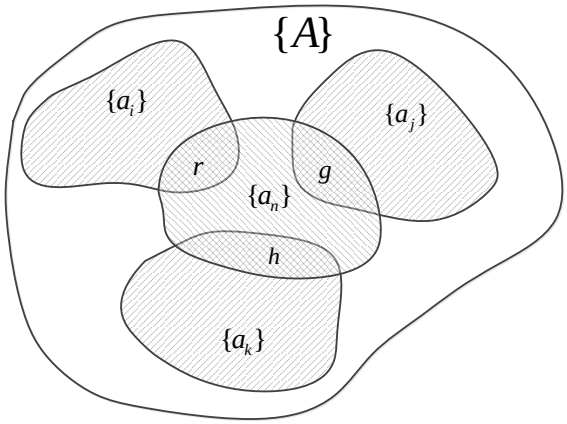

Рисунок 2 – Множина обсягу робіт з аналізу предметної області і формулювання вимог

Множина {А} у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$A = \{a_i, a_j, a_k, a_n\}; \; a_i = \overline{1,\alpha}, \; a_j = \overline{1,\beta}, \; a_k = \overline{1,\gamma}, \; a_n = \overline{1,\delta}, \quad (10)$$

де: α – обсяги робіт із оцінки вимог та потреб стейкхолдерів; β – обсяги робіт з визначення обмежень та вимог до функціональності;

γ – обсяги робіт з визначення фінансових та ресурсних обмежень проєкту;

δ – обсяги робіт із створення документації проєкту.

В множині {А} між підмножинами існують як операції об'єднання, так і перетину:

$$a_i \cap a_n; a_i \cup a_j; a_i \cup a_k; a_j \cap a_n; a_j \cup a_i; a_j \cup a_k; a_k \cap a_n; a_k \cup a_i; a_k \cup a_j. \quad (11)$$

Усі операції об'єднання є комутативними, тому узагальнюючи вираз можна записати:

$$A \supseteq \bigcup_{i=1}^{3} (a_i, a_j, a_k)., \quad (12)$$

Проте даний вираз не враховує операцій перетину множин із множиною a_n. Зокрема:

$$a_i \cap a_n = \{r : r \in a_i \; i \; r \in a_n\}, \quad (13)$$

де: r – сформоване технічне завдання проєкту;

$$a_j \cap a_n = \{g : g \in a_j \; i \; g \in a_n\}, \quad 14)$$

де: g – сформована специфікація програмного забезпечення;

$$a_k \cap a_n = \{h : h \in a_k \; i \; h \in a_n\}, \quad (15)$$

де: h – сформований фінансовий та ресурсний план проєкту.

Наступний етап життєвого циклу програмного забезпечення – проєктування архітектури програми та UX-дизайну інтерфейсу. На рисунку 3 зображена графічна модель цього етапу у вигляді множини {В}.

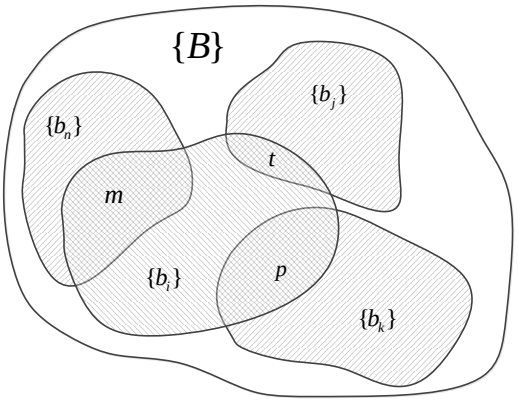

Рисунок 3 – Множина обсягу робіт з проєктування архітектури програми та UX-дизайну

Множина $\{B\}$ у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$B=\{b_i,b_j,b_k,b_n\};\ b_i=\overline{1,\varepsilon},\ b_j=\overline{1,\eta},\ b_k=\overline{1,\theta},\ b_n=\overline{1,\lambda}, \quad (16)$$

де: ε – обсяги робіт із створення архітектури системи;

η – обсяги робіт із вибору програмних технологій;

θ – обсяги робіт із побудови та написання алгоритмів;

λ – обсяги робіт з розробки дизайну та моделі людино-машинної взаємодії.

В множині $\{B\}$ між підмножинами також існують як операції об'єднання, так і перетину:

$$b_i\cap b_j;b_j\cup b_k;b_j\cup b_n;b_n\cap b_i;b_n\cup b_k;b_n\cup b_j;b_k\cap b_i;b_k\cup b_j;b_k\cup b_n. \quad (17)$$

Усі операції об'єднання є комутативними, тому узагальнюючи вираз можна записати:

$$B\supseteq\bigcap_{i=1}^{3}(b_j,b_k,b_n)., \quad (18)$$

Проте цей вираз не враховує операцій перетину множин із множиною b_i. Зокрема:

$$b_j\cap b_i=\{t:t\in b_j\ i\ t\in b_i\}, \quad (19)$$

де: t – архітектура структурних елементів програмної системи у вигляді діаграми класів та діаграми об'єктів;

$$b_n\cap b_i=\{m:m\in b_n\ i\ t\in b_i\}, \quad (20)$$

де: m – архітектура структурних елементів програмної системи у вигляді діаграми прецедентів, станів та компонентів (прототип програми);

$$b_k\cap b_i=\{p:p\in b_k\ i\ t\in b_i\}, \quad (21)$$

де: p – архітектура структурних елементів програмної системи у вигляді діаграм взаємодії та розміщення.

Наступний етап життєвого циклу програмного забезпечення – реалізація програмного продукту в кодах (імплементація програмної системи). На рисунку 4 зображена геометрична модель даного етапу у вигляді множини $\{C\}$.

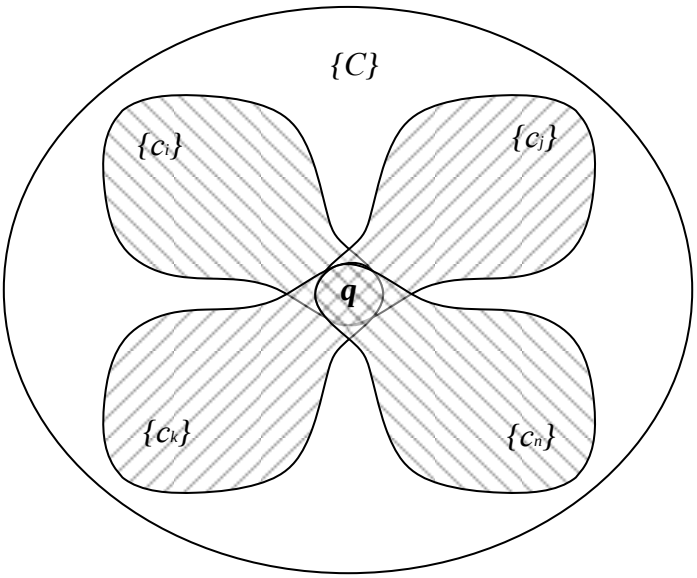

Рисунок 4 – Множина обсягу робіт з реалізації програмної системи в кодах

Множина $\{C\}$ у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$C=\{c_i,c_j,c_k,c_n\};\ c_i=\overline{1,\mu},\ c_j=\overline{1,\nu},\ c_k=\overline{1,\pi},\ c_n=\overline{1,\rho}, \quad (22)$$

де: μ – обсяги робіт з реалізації програмної системи в кодах;

ν – обсяги робіт з оптимізації та тестування коду (Unit tests);

π – обсяги робіт з виявлення та усунення дефектів, що виникають під час тестування (Unit Tests);

ρ – обсяги робіт із розробки документації до коду програми.

В множині $\{C\}$ між підмножинами існують виключно операції перетину, оскільки всі її підмножини максимально залежні одна від іншої:

$$c_i \cap c_j \cap c_k \cap c_n = \{q : q \in c_i\ i\ q \in c_j\ i\ q \in c_k\ i\ q \in c_n\}, \quad (23)$$

де: q – готовий програмний модуль із супровідною документацією.

Узагальнено взаємозв'язки множини $\{C\}$ можна представити так:

$$C \supseteq \bigcap_{i=1}^{4}(c_i, c_j, c_k, c_n)., \quad (24)$$

Черговий етап життєвого циклу програмного забезпечення – тестування програмного продукту (QC/QA). На рисунку 5 зображена геометрична модель цього етапу у вигляді множини $\{D\}$.

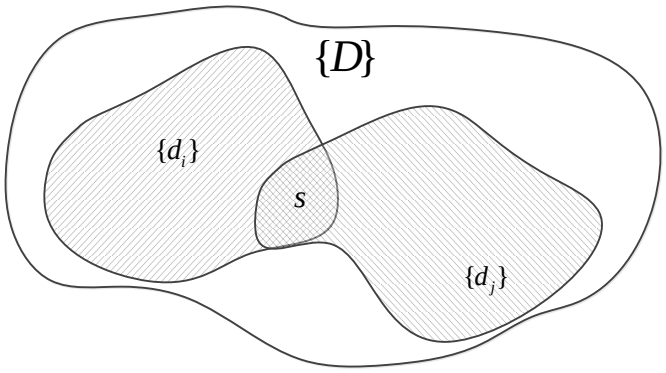

Рисунок 5 – Множина обсягу робіт із тестування програмної системи

Множина $\{D\}$ у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$D=\{d_i,d_j\};\ d_i=\overline{1,\sigma},\ d_j=\overline{1,\tau},, \quad (25)$$

де: σ – обсяги робіт із виконання тестів на функціональність, продуктивність, безпеку тощо;

τ – обсяги робіт із перевірки відповідності вимогам технічного завдання.

В множині $\{D\}$ між підмножинами існують виключно операції перетину, оскільки всі її підмножини максимально залежні одна від одної:

$$d_i \cap d_j = \{s : s \in d_i\ i\ s \in d_j\}, \quad (26)$$

де: s – пройдені тест-кейси (test cases) або/та звіти про помилки (bug reports).

По завершенні етапу тестування життєвий цикл програмного забезпечення переходить до етапу впровадження програмного продукту шляхом його розгортання на стороні клієнта (видача замовнику). На рисунку 6 відтворена геометрична модель цього етапу у вигляді множини $\{E\}$.

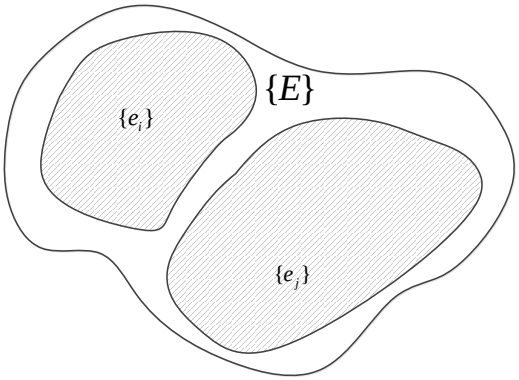

Рисунок 6 – Множина обсягу робіт із впровадження програмної системи

Множина $\{E\}$ у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$E = \{e_i, e_j\};\ e_i = \overline{1,\upsilon},\ e_j = \overline{1,\varphi}, \quad (27)$$

де: υ – обсяги робіт з підготовки програмного середовища для розгортання програмного продукту;

φ – обсяги робіт із навчання користувачів.

В множині $\{E\}$ між підмножинамиіснують виключно операції об'єднання і оскільки ці операції є комутативними, їх можна описати так:

$$E \supseteq \bigcup_{i=1}^{2}(e_i, e_j).., \quad (28)$$

І останній етап життєвого циклу програмного забезпечення – супровід програми під час експлуатації (підтримка програмного забезпечення) до моменту завершення експлуатації продукту. На рисунку 7 зображена геометрична модель цього етапу у вигляді множини $\{F\}$.

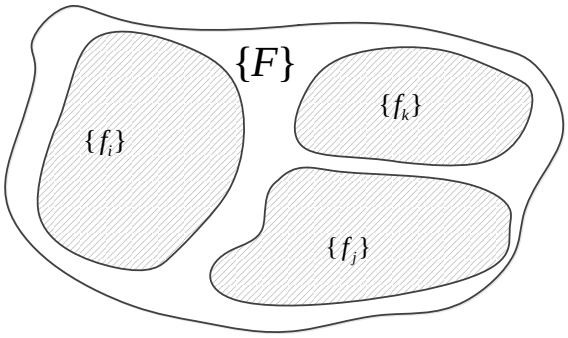

Рисунок 7 – Множина обсягу робіт із супроводу програмної системи під час експлуатації.

Множина $\{F\}$ у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$F = \{f_i, f_j, f_k\};\ f_i = \overline{1,\psi},\ f_j = \overline{1,\omega}\ f_k = \overline{1,\chi}, \quad (29)$$

де: ψ – обсяги робіт із виправлення помилок та виявлення нових дефектів;

ω – обсяги робіт із надання користувачам необхідної допомоги та підтримка;

χ – обсяги робіт із доповнення програмного продукту новим функціоналом, що концептуально відтворює усі попередньо описані етапи ЖЦ програмних систем

В множині $\{F\}$ між підмножинами існують виключно операції об'єднання і оскільки ці операції є комутативними, можна цю множину таким чином:

$$F \supseteq \bigcup_{i=1}^{3}(f_i, f_j, f_k).., \quad (30)$$

Представлені геометричні та математичні моделі множин із використанням понятійного апарату теорії множин допомагають узагальнити структуру та описати усі етапи життєвого циклу програмного забезпечення на різних етапах (аналіз предметної області та формулювання системних вимог, проєктування архітектури програми та UX-дизайну інтерфейсу, реалізація програми в кодах, тестування програми, впровадження програми, супровід програми під час експлуатації).

Обговорення результатів досліджень. Відомі на сьогодні дослідження в царині менеджменту життєвого циклу програмних систем використовують лише теоретичні відомості про етапи розробки нового програмного продукту і не вивчають можливостей їх автоматизації. Зважаючи на актуальну науково-прикладну задачу автоматизації окремих процесів управління життєвим циклом програмних систем, ми провели детальний аналіз та моделювання усіх етапів розробки програмного продукту із використанням понятійного апарату теорії множин. Представлена в статті математична модель дає підстави сформувати повну та чітку уяву про обсяги робіт, виконання яких лежить в основі успішної реалізації програмних систем, у тому числі безпеко-орієнтованого спрямування. Отримана модель дає охарактеризувати окремі етапи розробки продукту та аналітично описати життєвий цикл програмного забезпечення, що є передумовою для пошуку шляхів автоматизації окремих процесів підтримки прийняття управлінських рішень.

Висновки. За результатами проведених теоретичних досліджень можна зробити такі висновки: шляхом моделювання процесу розробки програмних систем із використанням понятійного апарату теорії множин отримано математичну модель їх життєвого циклу, яка дає змогу охарактеризувати обсяги робіт на окремих етапах створення програмного продукту, встановити взаємозв'язки та взаємозалежності між ними та сформувати фундаментальні принципи автоматизації процедури підтримки прийняття управлінських рішень на різних етапах створення програмних систем, зокрема безпеко-орієнтованого спрямування.

Список літератури:

1. Алексієв В. О., Труш О. М. Аналіз методів та оптимізація процесу розробки програмних продуктів. Моделювання та інформаційні технології в науці, техніці та освіті: зб. наук. праць міжнар. наук.-практ. Internet-конф. 21-22 лист. 2018 р. Харків, 2018. С. 45-51.

2. Singh B. and Gautam S. The Impact of Software Development Process on Software Quality: A Review. 8th International Conference on Computational Intelligence and Communication Networks (CICN). Tehri, India, 2016. pp. 666-672, https://doi.org/10.1109/CICN.2016.137

3. Кузь М. В, Пікуляк М. В. Остафійчук Т. Д. Модель системи управління якістю процесу розробки програмного забезпечення. Proceedings of the 2019 Scientific Seminar on Innovative Solutions in Software Engineering. Івано-Франківськ, 2019. С. 19-21, https://doi.org/10.5281/zenodo.4091480

4. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розробки програмного забезпечення. Системи управління, навігації та зв'язку. Полтава, 2018. Т. 3(49). С. 116-125. https://doi.org/10.26906/SUNZ.2018.3.116

5. Ribeiro S., Schmitz E., Alencar A., Silva M. Literature Review on the Theory of Constraints Applied in the Software Development Process. IEEE Latin America Transactions. 2018. vol. 16, no. 11, pp. 2747-2756,
https://doi.org/10.1109/TLA.2018.8795116

6. Кордунова Ю. С., Смотр О. О., Кокотко І. Я., Малець Р. Б. Аналіз традиційного та гнучкого підходів до створення програмного забезпечення в динамічних умовах. Управління розвитком складних систем. Київ, 2021. № 47. С. 71 – 77, https://doi.org/10.32347/2412-9933.2021.47.71-77

7. Kordunova Y., Prydatko O., Smotr O., Golovatyi R. Expert Decision Support System Modeling in Lifecycle Management of Specialized Software. Lecture Notes on Data Engineering and Communications Technologies, Springer, Switzerland. Vol. 149, 2022, pp. 367-383, https://doi.org/10.1007/978-3-031-16203-9_22

8. Kuhrmann M. et al., Hybrid Software Development Approaches in Practice: A European Perspective. IEEE Software. 2019. vol. 36, no. 4, pp. 20-31, https://doi.org/10.1109/MS.2018.110161245

9. Jain P., Sharma A., Ahuja L. The Impact of Agile Software Development Process on the Quality of Software Product. 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). Noida, India, 2018, pp. 812-815, https://doi.org/10.1109/ICRITO.2018.8748529

10. Bertling M., Caroli H., Dannapfel M., Burggraf P. The minimal viable production system (mvps) – an approach for agile (automotive) factory planning in a disruptive environment. Advances in Production Research. Springer International Publishing. 2019. pp. 24–33,
https://doi.org/10.1007/978-3-030-03451-1_3

11. Ковальчук О. І., Зачко О. Б. Моделі життєвого циклу розвитку проєктних команд в системі цивільного захисту. Вісник Львівського державного університету безпеки життєдіяльності. Львів, 2022. №25. С. 71-78.
https://doi.org/10.32447/20784643.25.2022.08

12. Malets I., Prydatko O., Popovych V., Dominik A. Interactive Computer Simulators in Rescuer Training and Research of their Optimal Use Indicator. 2018 IEEE Second Conference on Data Stream Mining & Processing. Lviv, 2018. – №2 – 558-562.

13. Придатко О. В., Придатко В. В., Борзов Ю. О., Дзень В. Є. Інтеграція новаційного методу мобільного навчання в освітні проекти підготовки розробників програмного забезпечення. Вісник ЛДУБЖД, Львів: ЛДУ БЖД, 2018. – №18. – С.70-80.

14. Barraood S. O., Mohd H., Baharom F. A Comparison Study of Software Testing Activities in Agile Methods. Knowledge Management International Conference (KMICe) Virtual Conference. Malaysia, 2021. pp. 130-137

15. Асєєва А. В., Кулаковська І. В. Аналіз проблем вибору технології для розробки програмного забезпечення. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. Луцьк, 2019. №37. С. 10 – 18,
https://doi.org/10.36910/6775-2524-0560-2019-37-2

16. Barbareschi M., Barone S., Carbone R. et al. Scrum for safety: an agile methodology for safety-critical software systems. Software Qual J. 2022. №30, pp. 1067–1088
https://doi.org/10.1007/s11219-022-09593-2

17. Thesing T., Feldmann C., Burchardt M. Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project. Procedia Computer Science. 2021. № 181. pp. 746 – 756.

18. Hovorushchenko T., Herts A.; Hnatchuk Y. Concept of Intelligent Decision Support System in the Legal Regulation of the Surrogate Motherhood. International Workshop on Informatics & Data-Driven Medicine, 2019, pp: 57-68.

References:

1. Aleksiiev, V. O., Trush, O. M., Alekseev, V. O. and Trush, E. N. (2018). "Analysis of methods and optimization of the software development process", Modeliuvannia ta informatsiini tekhnolohii v nautsi, tekhnitsi ta osviti, Kharkiv, November 21-22, 2018, pp. 45-51. (in Ukr.).

2. Singh, B. and Gautam, S. (2016). "The Impact of Software Development Process on Software Quality: A Review", 8th International Conference on Computational Intelligence and

Communication Networks (CICN), Tehri, India, 2016. pp. 666-672, doi: https://doi.org/10.1109/CICN.2016.137

3. Kuz, M. V, Pikuliak, M. V. and Ostafiichuk, T. D. (2019). "The model of the quality management system of the software development process", Proceedings of the 2019 Scientific Seminar on Innovative Solutions in Software Engineering, Ivano-Frankivsk, December 10, 2019. 19-21, doi: https://doi.org/10.5281/zenodo.4091480

4. Kovalenko, O. V. (2018). "Methods of qualitative analysis and quantitative assessment of software development risks", Systemy upravlinnia, navihatsii ta zv'iazku., 3(49), 116-125. doi.org/10.26906/SUNZ.2018.3.116

5. Ribeiro, S., Schmitz, E., Alencar, A. and Silva, M. (2018) "Literature Review on the Theory of Constraints Applied in the Software Development Process", IEEE Latin America Transactions, 16, 11, 2747-2756, https://doi.org/10.1109/TLA.2018.8795116

6. Kordunova, Yu., Smotr, O., Kokotko, I. and Malets, R. (2021). "Analysis of the traditional and flexible approaches to creating software in dynamic conditions", Management of Development of Complex Systems, 47, 71–77, dx.doi.org\10.32347/2412-9933.2021.47.71-77.

7. Kordunova, Y., Prydatko, O., Smotr, O. and Golovatyi, R. (2023). "Expert Decision Support System Modeling in Lifecycle Management of Specialized Software", Lecture Notes on Data Engineering and Communications Technologies, Springer, Switzerland, 149, 367-383, https://doi.org/10.1007/978-3-031-16203-9_22

8. Kuhrmann, M. et al. (2019) "Hybrid Software Development Approaches in Practice: A European Perspective", IEEE Software, 36, 4, 20-31, https://doi.org/10.1109/MS.2018.110161245

9. Jain, P., Sharma, A. and Ahuja, L. (2018) "The Impact of Agile Software Development Process on the Quality of Software Product", 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 812-815, https://doi.org/10.1109/ICRITO.2018.8748529

10. Bertling, M., Caroli, H., Dannapfel, M. and Burggraf, P. (2019) "The minimal viable production system (mvps) – an approach for agile (automotive) factory planning in a disruptive environment", Advances in Production Research. Springer International Publishing. 24–33, https://doi.org/10.1007/978-3-030-03451-1_3

11. Kovalchuk, O. I., and Zachko, O. B. (2022). "Life cycle models of project teams development in the civil protection system". Bulletin of Lviv State University of Life Safety, 25, 71-78. https://doi.org/10.32447/20784643.25.2022.08

12. Malets, I., Prydatko, O., Popovych, V. and Dominik, A. (2018). "Interactive Computer Simulators in Rescuer Training and Research of their Optimal Use Indicator". 2018 IEEE Second Conference on Data Stream Mining & Processing. Lviv, 2, 558-562.

13. Prydatko, O. V., Prydatko, V. V., Borzov, Yu. O. and Dzen V. Ye. (2018). "Integration of the new method of mobile education in educational projects of programmer training". Bulletin of Lviv State University of Life Safety, 18, 70-80.

14. Barraood, S. O., Mohd, H. and Baharom, F. A. (2021). "Comparison Study of Software Testing Activities in Agile Methods" Knowledge Management International Conference (KMICe),130-137

15. Asieieva, A., and Kulakovska, I. (2019). "Analysis of technology selection problems for software development". Computer-integrated technologies: education, science, production, 37, 10-18. https://doi.org/10.36910/6775-2524-0560-2019-37-2

16. Barbareschi, M., Barone, S., Carbone, R. et al. (2022). "Scrum for safety: an agile methodology for safety-critical software systems". Software Qual J, 30, 1067–1088 doi.org/10.1007/s11219-022-09593-2

17. Thesing, T., Feldmann, C. and Burchardt, M. (2021). "Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project". Procedia Computer Science, 181, 746 – 756.

18. Hovorushchenko, T., Herts, A. and Hnatchuk, Y. (2019). "Concept of Intelligent Decision Support System in the Legal Regulation of the Surrogate Motherhood". International Workshop on Informatics & Data-Driven Medicine, 57-68.

Науково-методична стаття.
Надійшла до редакції 12.05.2023.
Прийнято до публікації 18.05.2023.