УДК 681.3

*В.А. Алексеєв, В.С. Терещенко*

# БАГАТОВАРІАНТНІСТЬ НОТАЦІЙ ПРЕДСТАВЛЕННЯ АЛГОРИТМІВ ФУНКЦІОНУВАННЯ ПРОГРАМНИХ ЗАСОБІВ ЯК ШЛЯХ ДО ЇХ ВЕРИФІКАЦІЇ

Розглядаються різні варіанти нотацій представлення алгоритмів прикладних задач у інформаційних системах, що дає змогу до їх програмної реалізації провести верифікацію прийнятих рішень щодо їх побудови, використовуючи для цього різноманітні засоби, притаманні кожній з цих нотацій.

## Вступ

Суспільство, структури, технічна база та людський потенціал якого пристосовані до оптимального перетворення знань у інформаційний ресурс та перероблення його з метою перекладання з пасивних форм (книги, статті, патенти тощо) у активні (моделі, алгоритми, програми, проекти), є інформаційним суспільством [1]. Саме на створення такого суспільства в нашій країні направлено дію Закону України "Про Національну програму інформатизації", тобто на створення сукупності взаємопов'язаних організаційних, правових, політичних, соціально-економічних, науково-технічних, виробничих процесів, що спрямовані на створення умов для задоволення інформаційних потреб громадян та суспільства на основі створення, розвитку і використання інформаційних систем, мереж, ресурсів та інформаційних технологій, які побудовані на основі застосування сучасної обчислювальної та комунікаційної техніки [2]. Інформатизація приходить в усі сфери життя суспільства. Важко собі уявити установу чи підприємство будь-якої галузі, де б не використовувалась обчислювальна техніка та інформаційні або інформаційно-телекомунікаційні системи. Їх проектування та створення, то справа відповідних фахівців: системних проектувальників та програмістів з залученням необхідного інструментарію та методів проектування. Але розвиток таких вже існуючих систем, нарощування їх функціональних можливостей за рахунок створення додаткових підсистем та програмно-технічних комплексів для вирішення нових прикладних задач не може обійтись без їхніх користувачів, що мають повноваження здійснювати обробку функціональної інформації цих систем для здійснення своїх функціональних обов'язків. Тільки такі спеціалісти, можуть визначити круг прикладних задач, які необхідно вирішувати в межах інформаційної системи для успішного функціонування їх установ або підприємств, визначити вимоги до їх функціонування та експлуатації.

Аналізу таких вимог розробниками підсистеми буває недостатньо для визначення шляхів реалізації (програмування) запропонованої прикладної задачі. Для створення програмного продукту для такої задачі необхідний чіткий алгоритм її реалізації. Тут і постає питання з опису алгоритмів – упорядкованих скінчених наборів чітко визначених правил для розв’язання задач за скінчену кількість кроків [3]. Користувачі майбутніх підсистем, хоча і не можуть створити досконалий алгоритм та описати його у вигляді блок-схеми, але вони мають достатній досвід для формулювання, постановки та вербального опису визначеної прикладної задачі в межах тактичного завдання.

У сучасних інформаційних технологіях фаза життєвого циклу, на якій фіксуються вимоги на розробку програмного забезпечення, є визначальною для його якості, термінів та вартості робіт. Саме на цій фазі має бути зафіксовано реальні потреби користувачів, що стосуються функціональних, операційних та сервісних можливостей, які має реалізувати розробник. Ціна помилок та нечітких неоднозначних формулювань користувача на цій фазі дуже висока, бо час і засоби витрачаються на непотрібний користува-

чеві (замовникові) програмний продукт, оскільки він мав на увазі зовсім інше, але не зумів сформулювати свої потреби. Внесення необхідних коректив при цьому може вимагати значних переробок, а інколи і повного перепроектування і, відповідно, перепрограмування [4].

Для уникнення такої негативної перспективи необхідно будь-який вербальний опис прикладної задачі, виконаний користувачем, поступово, за допомогою обумовлених дій розробника та користувача перетворити у завершену блок-схему алгоритму з повним набором коментарів, яка б повністю задовольнила потреби програміста і дозволила йому створити необхідний програмний продукт. Одним із шляхів для досягнення такої мети пропонується між згаданими етапами, які можна вважати описами алгоритму, використати проміжні описи, які з вже згаданими утворять упорядкований (направлений) ряд описів алгоритмів, можливості яких, з точки зору їх формалізації та жорсткості правил і вимог до їх побудови, послідовно збільшуються та поступово наближуються до блок-схемного опису. Таким чином, кожний з наступних у цьому ряді описів не зможе бути повністю створений, доки, з точки зору правил, прийнятих в такому описі, не будуть усунуті помилки та нечіткі формулювання у попередньому описі цього ряду. Така поступова поетапна верифікація описів дасть змогу послідовно вилучити вади з кожного з них. Такий підхід повністю відповідає концепції каскадної моделі, наведеної у [5, с. 67], життєвого циклу програмного продукту.

На даний час серед різноманітних описів або нотацій представлення алгоритмів функціонування програмних засобів підсистем в інформаційних системах, що застосовуються під час їх створення, можна виділити шість найбільш поширених способів описів, які утворюють вищезгаданий ряд:

- вербальний спосіб опису алгоритму;
- табличний спосіб опису алгоритму;
- граф-схемний спосіб опису алгоритму;
- формальний або математичний спосіб опису алгоритму;
- блок-схемний спосіб опису алгоритму;
- програмний спосіб опису алгоритму, тобто створення його програми.

Кожний з цих підходів має свої переваги та недоліки, свою ступінь складності та формалізації, свої можливості щодо повноти опису та деталізації процесів, процедур, операцій, дій, команд, що застосовано в них. Але всі вони, кожний окремо, не вільні від можливих помилок і нечітких формулювань розробників, та послідовна розробка описів цього ряду й узгодження кожного з них з попереднім описом, з обов'язковим залученням їх авторів-розробників, дозволяє максимально знизити кількість таких помилок. При цьому при розробці наступного опису часто можуть виникнути утруднення, які спонукатимуть потребу у внесенні змін до попереднього опису, і таким чином, поступово, використовуючи ітеративний підхід, здійснювати його поетапну верифікацію – перевірку системи, що здійснюється за допомогою формальних засобів для визначення відповідності системи установленим вимогам [6].

При проектуванні та розробленні програмних засобів верифікація полягає у процесі перевірки результатів певної діяльності для визначення відповідності вимогам, встановленим щодо цієї діяльності [7]. Саме цим проблемам – верифікації описів алгоритму на етапі проектування програмних засобів, тобто, на ранніх стадіях їх життєвого циклу [8, с. 46] на відміну від визначеної у літературних джерелах верифікації вимог до розробки та планів створення системи [4, 9] та верифікації програм вже створених програмних засобів [9, 10, 11], і присвячена дана робота.

## 1. Формальний опис структури узагальненого алгоритму

Яким би не був опис алгоритму, незалежно від способу його представлення, чи то у вигляді текстів, чи то у вигляді геометричної інтерпретації множин із зв'язками, що носять характер відображень,

він має бути ретельно структурованим. Таке структурування перш за все залежить від структури самого алгоритму – будь-якої скінченої системи правил перетворення інформації (даних) над будь-яким скінченим алфавітом [12, с. 38].

Зважаючи на те, що алгоритмами ще називають конструктивно завдані відповідності між словами в абстрактних алфавітах [13], вважатимемо, що наш узагальнений алгоритм $A$ обробки інформаційних ресурсів реалізовано у двох абстрактних алфавітах: скінченому наборі ($S^A$) його складових функціональних компонентів (ФК), призначених для обробки інформаційних ресурсів, та скінченому наборі ($T^A$) запам'ятовуючих компонентів сховища даних (КСД), призначених для зберігання результатів обробки як в короткостроковому (технологічному), так і в довгостроковому режимі.

Припустимо, що між словами першого з абстрактних алфавітів ($S^A$) узагальненого алгоритму $A$ конструктивно завдані відповідності щодо ієрархічності ФК – послідовної вкладеності процесів $B^A \in \{S^A\}$, процедур $C^A \in \{B^A\}$, операцій $D^A \in \{C^A\}$ та команд $E^A \in \{D^A\}$ так, що:

$$S^A = \bigcup_{i=1}^{I} B_i^A = \bigcup_{i=1}^{I}\bigcup_{j=1}^{J} C_{ij}^A = \bigcup_{i=1}^{I}\bigcup_{j=1}^{J}\bigcup_{k=1}^{K} D_{ijk}^A = \bigcup_{i=1}^{I}\bigcup_{j=1}^{J}\bigcup_{k=1}^{K}\bigcup_{l=1}^{L} E_{ijkl}^A$$

Треба зауважити, що опис ФК містить не тільки перелік дій у межах цих ФК (процеси, процедури, операції) та інформаційних ресурсів для їх забезпечення, але і умови, при яких вони можуть мати місце та відбуватися, а команди – ФК, що характерний тільки для програмного способу опису алгоритму.

Припустимо також, що між словами другого з абстрактних алфавітів ($T^A$) узагальненого алгоритму ($A$) конструктивно завдані відповідності щодо отримання результатів функціонування згаданих процесів, процедур, операцій, команд та зберігання їх у ієрархічних КСД – послідовно вкладених базах даних (БД) $V^A \in \{T^A\}$, таблицях БД (ТБД) $W^A \in \{V^A\} \in \{T^A\}$ та їх записах $X^A \in \{W^A\} \in \{V^A\} \in \{T^A\}$, реквізитах ТБД $Y^A \in \{W^A\} \in \{V^A\} \in \{T^A\}$ та їх записах $Z^A \in \{Y^A\} \in \{W^A\} \in \{V^A\} \in \{T^A\}$ так, що:

$$T^A = \bigcup_{m=1}^{M} V_m^A = \bigcup_{m=1}^{M}\bigcup_{n=1}^{N} W_{mn}^A = \bigcup_{m=1}^{M}\bigcup_{n=1}^{N}\bigcup_{o=1}^{O} X_{mno}^A,$$

а також:

$$T^A = \bigcup_{m=1}^{M} V_m^A = \bigcup_{m=1}^{M}\bigcup_{n=1}^{N} W_{mn}^A = \bigcup_{m=1}^{M}\bigcup_{n=1}^{N}\bigcup_{p=1}^{P} Y_{mnp}^A =$$

$$= \bigcup_{m=1}^{M}\bigcup_{n=1}^{N}\bigcup_{p=1}^{P}\bigcup_{r=1}^{R} Z_{mnpr}^A,$$

де $Z_{mnpR}^A = X_{mnO}^A \cdot Y_{mnP}^A$ – максимально можлива кількість записів реквізитів визначеної ТБД.

Тоді будь-який опис такого алгоритму, незалежно від способу його представлення, має включати слова $s^A$ алфавіту $S^A = \{B^A, C^A, D^A, E^A\}$, слова $t^A$ алфавіту $T^A = \{V^A, W^A, X^A, Y^A, Z^A\}$ та алфавітні оператори $\Gamma^A s^A = t^A$, тобто функції що задають відповідність між словами вхідного ($S^A$) та вихідного ($T^A$) алфавітів цих операторів, інакше кажучи, встановлюють зв'язки між ними або відображують вхідне слово у вихідне на заданому рівні абстракції.

Таким чином, опис алгоритму $A$ у формалізованому вигляді можна представити як:

$$\Im^A = \bigcup_{f=1}^{F} \Gamma_f^A s^A = t^A \qquad (1)$$

при $s^A \in \{S^A\}$ та $t^A \in \{T^A\}$.

Це формальне представлення опису алгоритму в дуже загальному вигляді, але його структура накладає свій відбиток на будь-який спосіб опису (нотацію) конкретного алгоритму. Через такий загальний вигляд не враховувались особливості нотацій вищеперелічених способів представлення алгоритмів, інструментарію цих нотацій, норм та правил його застосування, але ці особливості будуть висвітлені у подальших розділах.

## 2. Короткий опис нотацій представлення алгоритму

Як вже згадувалось, існують різні способи опису алгоритму – структурної нотації його представлення, тобто, структурного, блок-схемного або текстового подання аспектів проектування структури програмних засобів з об'єктів, компонентів, їх інтерфейсів і взаємозв'язків [8, с. 30]. Серед них у даній роботі розглянемо: вербальний $\Im_1^A$, табличний $\Im_2^A$, граф-схемний $\Im_3^A$, формальний $\Im_4^A$, блок-схемний $\Im_5^A$ та програмний $\Im_6^A$ описи алгоритму $A$ та їх порівняльний аналіз.

**Вербальний спосіб опису алгоритму** ($\Im_1^A$) за функціональним навантаженням загалом тотожний експертній або концептуальній моделі, де наводиться формулювання і вербальна постановка (визначення) задачі, та дескриптивній моделі, де наводиться семантичний опис задачі, а визначення задачі, як відомо з державного стандарту [6], є її формулювання, що може містити опис даних, а також метод, процедури і алгоритм її розв'язання.

До переваг вербального способу опису алгоритму належить застосування мови опису, яка дуже наближена до звичної людської мови, і, в залежності від вибраної нотації, крім застосування необхідної термінології визначеної предметної області, може включати деякі формалізми, наприклад, у позначенні різних видів інформаційних ресурсів: баз даних, їх таблиць та, можливо, їх реквізитів. Це дозволяє користувачеві, незнайомому з програмістським мистецтвом, зрозуміло викласти своє бачення щодо вирішення прикладних задач інформаційної системи у вигляді опису алгоритму звичною йому нотацією. У даному разі нотації визначатимуть міру узагальненості або, навпаки, – міру деталізації прийнятого способу опису.

До недоліків вербального способу опису слід віднести велику складність для програмістів визначити та обумовити склад обчислювальних процедур та операцій для досягнення результату та не завжди чітке визначення переліку і місць знаходження вхідних та вихідних даних у таких процедурах. Недоліком такого способу опису алгоритму є також нечітке визначення послідовності здійснення операцій у кожному з процесів та процедур, не завжди чітке визначення умовних переходів, наприклад: "якщо <умова>, то <перехід 1>" забуваючи при цьому, що обов'язково має бути "інакше <перехід 2>" тощо. Це може змусити під час опрацювання наступних описів повернутися до цього опису для його верифікації та висвітлення зазначених питань.

З енциклопедичного видання відомо, що таблиця це перелік відомостей, числових даних, приведених у певну систему та рознесених по графах [14]. Основним положенням такої системи для запропонованої нотації **табличного способу опису алгоритму** ($\Im_2^A$) є використання двобічної таблиці опису алгоритму (ТОА), лівий бік якої призначено для вхідного алфавіту $S^A$, а правий – для вихідного алфавіту $T^A$. Зрозуміло, що відповідність слів цих алфавітів у ТОА має забезпечувати алфавітний оператор $\Gamma^A s = t$. З лівого боку такої ТОА мають бути передбачені графи для опису ФК в залежності від ступеня її деталізації і умов функціонування ФК, включаючи вхідні дані та місця їх зберігання у КСД. З правого боку такої ТОА мають бути передбачені графи для зазначення місць зберігання у КСД результатів функціонування ФК. Такі ТОА, завдячуючи своїй структурі, навіть на початковому етапі проектування підсистеми, спонукають визначити не тільки ФК та умови і результати їх функціонування, але і необхідну структуру КСД (БД, ТБД, реквізити) для зберігання вхідних даних та результатів функціонування ФК.

Практика показала, що навіть невеликий, з точки зору сучасних інформаційних систем, алгоритм потребує для свого опису значної ТОА, яка надто незручна для огляду та аналізу. До того ж велика кількість чарунок (місць перетину граф та рядків) її правого боку часто

залишаються незаповненими, що свідчить про невдалу структуру такої таблиці. Тому таку ТОА слід модифікувати: згрупувати її рядки по окремих модифікованих таблицях опису алгоритму (МТОА) в залежності від ФК, тобто для кожного процесу, процедури або, навіть, операції має бути створена окрема МТОА, як це показано у табл. 1 для процедури $C_{ij}^{A}$ (*j*-ої процедури *i*-го процесу). Другий етап модифікації ТОА стосується її правого боку, де мають бути згруповані та рознесені по окремих МТОА графи, присвячені окремим КСД – БД або, навіть, ТБД в залежності від кількості реквізитів у цих ТБД, що задіяні для зберігання результатів функціонування визначеного ФК цієї МТОА, як це показано у табл. 1 для ТБД $W_{mn}$ (*n*-ої ТБД *m*-ої БД) та її записів і реквізитів $Y_{mnp'}$, $Y_{mnp''}$, $Y_{mnp'''}$ та $Y_{mnp''''}$ при $1 \le p \le P$, $1 \le p' \le P$, $1 \le p'' \le P$, $1 \le p''' \le P$, $1 \le p'''' \le P$ та $p \ne p' \ne p'' \ne p''' \ne p''''$.

Якщо результати операцій процедури $C_{ij}^{A}$ можуть зберігатись і в інших ТБД, то для них необхідно створювати такі ж за структурою, але окремі МТОА саме для цих ТБД.

До переваг табличного способу опису алгоритму, крім їхньої властивості до спонукання визначення структури КСД, можна віднести також застосування звичної людської мови, хоча таблична структура обумовлює досить жорсткі вимоги до викладення положень у такому описі цією мовою, що відбивається на підвищенні рівня гарантій щодо безпомилковості створення такого опису. Тут прозоріше виглядають каузальні зв'язки, які спонукають наводити у визначених графах лівого боку МТОА тільки зміст процедур, їх операцій, вхідних даних та умов, при яких вони мають відбуватись у відповідності до вхідного алфавіту $S^{A}$. У графах правого боку МТОА безперечно мають бути тільки результати цих операції та КСД, де вони зберігаються, у відповідності до вихідного алфавіту $T^{A}$. Таким чином, саме на цьому етапі, при цьому способі опису алгоритму проходить межа взаємних узгоджень між користувачем та системним проектувальником на будь-якому етапі верифікації.

Недоліком табличного способу опису алгоритму є відсутність конструктивно в ньому закладених жорстких вимог до визначення послідовності проведення операцій у кожному з процесів та процедур. Тому це більшістю залежить від кваліфікації, прискіпливості та досвідченості проектувальника, що не дає безперечних гарантій на всебічну безпомилковість такого опису.

При **граф-схемному способі опису алгоритму** ($\Im_3^A$), для зображення його структури використовується граф-схема, що представляє собою скінчену множину з'єднаних між собою вершин у вигляді геометричних фігур, що звуться вузлами граф-схеми [13]. На визначеному рівні деталізації опису (для процесів, процедур, операцій) у якості вузлів застосовуються спеціалізовані двографові таблички – табели (ікони), в яких крім назви ТБД зазначеного КСД у лівій графі колонкою наведено перелік реквізитів для поточного (старого) запису ТБД, а у правій – нового запису. Таким чином, сама структура такого способу опису обумовлює чітке визначення як вхідних даних для проведення операцій (старі записи) та КСД для їх зберігання, так і їх результатів (нові записи) та відповідних КСД. Це є перевагою такого способу опису алгоритму. Як ребра такої граф-схеми застосовуються операції, що зображуються колами чи овалами, зв'язаними стрілками з реквізитами у табелах, пронумеровані (у відповідності до послідовності їх здійснення) та позначені (відповідними абревіатурами, прийнятими у вербальному та табличному описах) або тільки стрілками, якщо операцією є тільки присвоєння значення даним або їх пересилання. Якщо до цих граф-схем застосувати розвинутий математичний апарат теорії графів, скажімо, для їх еквівалентного перетворення з одночасною мінімізацією кількості вузлів, то виникає перспектива до оптимізації структури КСД.

Таблиця 1. Структура МТОА для ФК-$C_{ij}^{A}$ визначеного процесу $B_i^A$ та ТБД-$W_{mn}$

| **Вхідний алфавіт** $S^A$ **алгоритму** *A* | | **Вихідний алфавіт** $T^A$ **алгоритму** *A* | | | | |
|---|---|---|---|---|---|---|
| **ФК процесу** $B_i^A$ | | **Результати операції над записами та реквізитами ТБД** $W_{mn}$ | | | | |
| Процедура та її умови | Операції процедури та їх умови | Записи ТБД $W_{mn}$ | Реквізит $Y_{mnp'}$ ТБД $W_{mn}$ | Реквізит $Y_{mnp''}$ ТБД $W_{mn}$ | Реквізит $Y_{mnp'''}$ ТБД $W_{mn}$ | Реквізит $Y_{mnp''''}$ ТБД $W_{mn}$ |
| $C_{ij}^{A}$ | $D_{ij1}^{A}$ | $f_1^A\left(D_{ij1}^A, W_{mn}\right)$ | $f_1^A\left(D_{ij1}^A, Y_{mnp'}\right)$ | $f_1^A\left(D_{ij1}^A, Y_{mnp''}\right)$ | $f_1^A\left(D_{ij1}^A, Y_{mnp'''}\right)$ | $f_1^A\left(D_{ij1}^A, Y_{mnp''''}\right)$ |
| | $D_{ij2}^{A}$ | $f_2^A\left(D_{ij2}^A, W_{mn}\right)$ | $f_2^A\left(D_{ij2}^A, Y_{mnp'}\right)$ | $f_2^A\left(D_{ij2}^A, Y_{mnp''}\right)$ | $f_2^A\left(D_{ij2}^A, Y_{mnp'''}\right)$ | $f_2^A\left(D_{ij2}^A, Y_{mnp''''}\right)$ |
| | ... | ... | ... | ... | ... | ... |
| | $D_{ijk}^{A}$ | $f_k^A\left(D_{ijk}^A, W_{mn}\right)$ | $f_k^A\left(D_{ijk}^A, Y_{mnp'}\right)$ | $f_k^A\left(D_{ijk}^A, Y_{mnp''}\right)$ | $f_k^A\left(D_{ijk}^A, Y_{mnp'''}\right)$ | $f_k^A\left(D_{ijk}^A, Y_{mnp''''}\right)$ |
| | ... | ... | ... | ... | ... | ... |
| | $D_{ijK}^{A}$ | $f_K^A\left(D_{ijK}^A, W_{mn}\right)$ | $f_K^A\left(D_{ijK}^A, Y_{mnp'}\right)$ | $f_K^A\left(D_{ijK}^A, Y_{mnp''}\right)$ | $f_K^A\left(D_{ijK}^A, Y_{mnp'''}\right)$ | $f_K^A\left(D_{ijK}^A, Y_{mnp''''}\right)$ |

Недоліком такого способу є слабкість конструктивно закладеної вимоги до визначення послідовності операцій у кожному з процесів. У відповідності до неї, хоча і застосовується нумерація операцій, але вона не настільки жорстка, щоб забезпечити впевненість у безпомилкових діях розробника. І все таки, на відміну від табличного способу, можливість визначення послідовності операцій при граф-схемному способі більше виражена.

**Формальний спосіб опису алгоритму** ($\Im_4^A$) за функціональним навантаженням тотожний математичній моделі – системі математичних виразів, що описують характеристики об'єкта моделювання та взаємозв'язки між ними. При його створенні використовується опис математичними методами окремих операцій для встановлення кількісних та логічних залежностей між різними елементами ФК у межах цих операцій. Перевагами такого способу опису є закладена в ньому конструктивно вимога до визначення послідовності проведення операцій у кожному з процесів та процедур (жорстка нумерація операцій).

Недоліком такого способу є дуже жорстка лаконічність у представленні та описі операцій. Тому його доцільно використовувати разом з іншими описами.

При **блок-схемному способі опису алгоритму** ($\Im_5^A$), найбільш поширеному в програмістській практиці, для зображення його структури використовується блок-схема – графічне подання алгоритму чи задачі за допомогою спеціальних символів для проведення їхнього аналізу або розв'язку [15]. У такій блок-схемі на визначеному рівні деталізації (для процесів, процедур, операцій, команд) застосовуються блоки у вигляді геометричних фігур, кожна з яких несе відповідне функціональне навантаження [16], блокам присвоюються номери, і вони споряджаються пояснювальним текстом. Напрям процесу обробки у блок-схемі позначається відповідними стрілками. Якщо один блок передає керування іншим блокам у залежності від виконання певних умов, то на стрілках позначається умова, при якій процес розгалужується.

Перевагами такого способу опису алгоритму є те, що він дає найбільш повне представлення програмісту про структуру алгоритму, необхідні дії, пов'язані з безпосередньою розробкою програми для його реалізації. Це пояснюється тим, що вже складено найбільш наближений до програми формальний опис процесу вирішення прикладної задачі з дуже чітким визначенням послідовності проведення операцій (*D*), а, можливо, і команд (*E*) у кожному з процесів (*B*) та процедур (*C*). Перевагами такого способу є також те, що він полегшує читання та розуміння вже створених програм для їх валідації та

верифікації, що зменшує кількість помилок при програмуванні [13].

Недоліком такого способу опису алгоритму є неодмінна необхідність у зусиллях системного проектувальника, який міг би переосмислити будь-який попередній опис у таку блок-схему, а також:

- слабка залежність відчуття завершеності опису від визначення місць зберігання у КСД проміжних та кінцевих результатів функціонування ФК (це, як правило, має зазначатися у коментарях, які далеко не завжди присутні на блок-схемі), що робить дуже критичним опис від кваліфікації та досвіду розробника, тобто можливе неповне визначення вихідного алфавіту оператора алгоритму;
- відсутність можливості встановлення певного ступеня деталізації, яка у подальшому необхідна при розробці програми, що призводить до опису одних блоків з надлишковими подробицями, а інших – у загальних рисах, через пряму залежність чіткості та повноти опису від кваліфікації та досвіду розробників.

Треба зазначити, що останній недолік притаманний всім попереднім способам опису алгоритму. Але на відміну від попередніх способів, він дає змогу встановити нечіткість у формулюванні пояснювального тексту у ФК попередніх описів, а особливо при визначенні розгалуження процесу обробки інформації в залежності від встановлених умов.

Вищеперелічені способи: табличний $\Im_2^A$, граф-схемний $\Im_3^A$, формальний $\Im_4^A$ та блок-схемний $\Im_5^A$ опису алгоритму за функціональним навантаженням тотожні різним аспектам логіко-інформаційної моделі, де наводиться опис алгоритму та бази даних [10, 17] і математичної (формальної) моделі. В описі $\Im_2^A$ приділяється більше уваги переліку ФК та їх описів, необхідних для визначення їх суті для отримання певних результатів; у описі $\Im_3^A$ – переліку місць зберігання вхідних даних для ФК та місць зберігання результатів їх функціонування у КСД; у описі $\Im_4^A$ – суті математичних дій у ході операцій; у описі $\Im_5^A$ – послідовності операцій та суті функціонування ФК . Саме ці розбіжності у наголосах цих описів і дозволяють здійснити всебічну верифікацію завершеного на цій стадії проекту перед переходом до програмування. За великим рахунком ці описи лише доповнюють вербальний опис, ретельно висвітлюючи його вузькі місця. Таким чином, можна сказати, що саме сукупність цих описів дає змогу створити довершений проект функціональної підсистеми – програмний опис її алгоритму.

**Програмний спосіб опису алгоритму** ($\Im_6^A$) за функціональним навантаженням тотожний командній моделі, де наводиться опис програми у нотації алгоритмічної мови або у нотації інструментальної системи розробки програмних засобів, наприклад: пакету S-Designer for Power Builder для проектування та генерації розподілених баз даних в архітектурі “клієнт-сервер” і створення функціональних модулів [18]. Машинні програми, створені за допомогою різноманітних трансляторів, описом алгоритму не вважатимемо, бо це вже є завершений програмний продукт, який в такому вигляді на змінних носіях розповсюджується серед користувачів, на відміну від алгоритмів, що є всього лише точно визначеним правилом дій, для якого задана вказівка, як і в якій послідовності це правило необхідно застосовувати до початкових даних задачі, щоб отримати її вирішення [19], тобто проектним документом.

До переваг програмного способу опису алгоритму необхідно віднести той факт, що такий завершений опис алгоритму за суттю є програмою – мовною конструкцією, що є впорядкованою послідовністю описів і команд, призначеною для оброблення даних [3], програмним засобом, який потребує тільки перевірки та налагоджування (валідація та верифікація програм). Перевагами такого способу є також дуже чітке визначення послідовності проведення команд (*E*), операцій (*D*) та процедур (*C*) у кожному з процесів (*B*), що і дозволяє функціонувати створеному програмному засобу.

Недоліком такого опису є те, що це найбільш складний спосіб опису алгоритму, що підвласний тільки безпосередньо програмісту, який, як правило, не є спеціалістом у предметній області, для автоматизації якої і призначена система. Навіть системний проектувальник тут безсилий, не кажучи вже про користувача.

Подальша трансляція програмного опису у машинну програму призводить до створення завершеного програмного продукту, після інсталяції якого на програмно-апаратних засобах утворюється фізична модель підсистеми.

## 3. Порівняльний аналіз описів (нотацій представлення) алгоритму

Зрозуміло що для того, щоб фізична модель функціонувала у відповідності до настанов, закладених у експертну, концептуальну, дескриптивну, логіко-інформаційну та формальну (математичну) моделі, вони мають бути еквівалентними у тому сенсі, що кінцеві результати їх реалізації мають відповідати повному набору настанов, закладених ще у початкову експертну модель, першим розробником якої у нашому випадку був користувач.

Відповідно до цього, для заданого алгоритму $A$ описи $\Im_1^A$, $\Im_2^A$, $\Im_3^A$, $\Im_4^A$, $\Im_5^A$ та $\Im_6^A$ теж мають бути еквівалентними. Описи (алгоритми) будемо вважати еквівалентними, якщо у них збігається функціональне навантаження (алфавітні оператори), але можуть не збігатися способи їх завдання. Фактично еквівалентність описів незалежно від їх вмісту (нотації) може бути підтверджена однаковим функціонуванням ($\Psi$) реалізованих за допомогою цих описів програмних засобів ($\Re(\Im^A)$). Тобто, умовою еквівалентності описів є:

$$\left[\Psi_1\left\langle\Re\left(\Im_1^A\right)\right\rangle \Leftrightarrow \Psi_2\left\langle\Re\left(\Im_2^A\right)\right\rangle \Leftrightarrow \Psi_3\left\langle\Re\left(\Im_3^A\right)\right\rangle \Leftrightarrow \right.$$
$$\left. \Leftrightarrow \Psi_4\left\langle\Re\left(\Im_4^A\right)\right\rangle \Leftrightarrow \Psi_5\left\langle\Re\left(\Im_5^A\right)\right\rangle \Leftrightarrow \Psi_6\left\langle\Re\left(\Im_6^A\right)\right\rangle\right] \Rightarrow$$
$$\Rightarrow \left[\Im_1^A \Leftrightarrow \Im_2^A \Leftrightarrow \Im_3^A \Leftrightarrow \Im_4^A \Leftrightarrow \Im_5^A \Leftrightarrow \Im_6^A\right]. \quad (2)$$

Інакше кажучи, функціональне навантаження незалежно від способу представлення цих еквівалентних описів має бути однаковим. Але на практиці не тільки способи представлення, але і функціональне їх навантаження дещо відрізняється через наявність у них нечітких формулювань та висловлень на початковій стадії створення цих описів – $\Im_1^{\circ A}$, $\Im_2^{\circ A}$, $\Im_3^{\circ A}$ та $\Im_4^{\circ A}$:

$$\Im_r^{\circ A} = \bigcup_{f=1}^{F} \Gamma_{rf}^{\circ A}\ s_r^{\circ A} = t_r^{\circ A} \quad (3)$$

при $s_r^{\circ A} \in \left\{S^A\right\}$ та $t_r^{\circ A} \in \left\{T^A\right\}$ для $r \in \left\{1, 2, 3, 4\right\}$, де $r$ – індекс способу опису алгоритму; $f \in \left\{1, 2, ..., F\right\}$ – індекс ФК, що розглядаються у даному описі.

Так, вербальний опис ($\Im_1^{\circ A}$) не дозволяє бути впевненим у його функціональній повноті, і може статися, що не всі необхідні для обробки інформації параметри завдані, не всі умови, які можуть бути у процесі її обробки, враховані та шляхи отримання результату передбачені, що неодмінно ускладнить розробку блок-схемного опису ($\Im_5$), якщо взагалі дозволить це зробити. Тут стануть у пригоді наступні способи: табличний ($\Im_2^{\circ A}$), граф-схемний ($\Im_3^{\circ A}$), який у якійсь мірі перегукується з широко відомим нині методом мови UML, та формальний ($\Im_4^{\circ A}$). Зрештою, описи $\Im_2^{\circ A}$, $\Im_3^{\circ A}$ та $\Im_4^{\circ A}$ і призначені саме для визначення недоопрацьованих питань, їхнього вирішення та удосконалення опису $\Im_1^{\circ A}$, а разом з ним і всіх інших описів для створення безпомилкового опису $\Im_5^A$. Такі способи дозволять певною мірою поступово у декілька заходів (циклічно) цілеспрямовано верифікувати не тільки вербальний опис алгоритму, а і табличний, граф-схемний та формальний і довести їх до потрібного стану, тобто до їх еквівалентності (функціональної тотожності $\Im_1$, $\Im_2^A$, $\Im_3^A$ та $\Im_4^A$), що дозволить впевнено перейти до еквівалентного з ними блок-схемного опису алгоритму ($\Im_5$), і, на ґрунті зазначених описів, створити еквівалентний ним програмний опис ($\Im_6$), який після трансляції стане працеспроможним програмним продуктом. Саме

еквівалентність наведених описів алгоритму дозволяє бути впевненим в успішному функціонуванні створеного за допомогою таких описів програмного продукту.

Таким чином, еквівалентність описів досягається у процесі їх верифікації внесенням відповідних, можливо, багаторазових поправок у початкові описи для їх коригування:

$$\Im_r^A = \Im_r^{\circ A} \bigcup_{f=1}^{F} \Pi_{rf}\left(\Gamma_{rf}^{\circ A}\ s_r^{\circ A} = t_r^{\circ A}\right). \qquad (4)$$

## 4. Поетапна ітеративна верифікація описів алгоритму різних нотацій

Наведені тут способи опису алгоритму застосовують як елементи парадигми імперативного програмування, де чітко визначається алгоритм та способи вирішення прикладної задачі (наприклад, способи опису $\Im_1^A$, $\Im_4^A$ та $\Im_5^A$), так і парадигми декларативного програмування, де чітко визначаються структури даних та властивість рішення задачі (наприклад, способи опису $\Im_2^A$ та $\Im_3^A$) [9]. Такий різнобічний підхід створює умови до щонайбільшого використання можливостей як одної, так і іншої парадигми. Зокрема, для проведення всебічної верифікації проекту алгоритму для перевірки, чи правильно він розробляється, шляхом дослідження вхідних робочих продуктів (описів попереднього способу в задекларованому ряді способів опису алгоритму) у вихідні робочі продукти (наступні описи).

Під час створення опису алгоритму вербальним способом ($\Im_1^{\circ A}$) одночасно користувачем здійснюється початкова його валідація у відповідності до загальних вимог реалізації цього алгоритму – специфікацій його проекту, які були обговорені та домовлені між замовником та розробником програмних засобів для вирішення прикладної задачі. Таку валідацію при створенні вербального опису не будемо розглядати серед етапів верифікації, які будуть викладені далі.

При створенні опису алгоритму табличним способом ($\Im_2^{\circ A}$) у системного проектувальника можуть виникнути питання щодо переліку та взаємодії ФК, переліку, напрямків та зв'язків інформаційних потоків з КСД запропонованої користувачем у вербальному описі алгоритму підсистеми на заданому ступені деталізації. Такий користувач разом з системним проектувальником має цілеспрямовано валідувати вербальний опис $\Im_1^{\circ A}$ та зняти ці питання шляхом внесення до нього змін та доповнень і отримати скоригований опис $\Im_1^{'A}$ (4). Згодом, ці спеціалісти мають проаналізувати та верифікувати табличний опис алгоритму $\Im_2^{\circ A}$ та шляхом внесення до нього необхідних змін і доповнень отримати скоригований опис $\Im_2^{'A}$ (4), узгоджуючи між собою ці два скоригованих описи. У ході такої валідації та верифікацій описів перевіряються відповідності потребам, несуперечності, повноти і можливості реалізації алгоритму, а також узгодженості зі стандартами програмної інженерії [8, с. 29]. Це **1-й етап верифікації** описів, у ході якого відбувається перше коригування вербального ($\Im_{r=1}^{\circ A}$) та табличного ($\Im_{r=2}^{\circ A}$) описів:

$$\Re_1 \Leftrightarrow \bigcup_{r=1}^{2}\left(\Im_r^{\circ A} \xrightarrow{\Pi_r^1} \Im_r^{'A}\right). \qquad (5)$$

Якщо при створенні опису алгоритму граф-схемним ($\Im_3^{\circ A}$) та формальним ($\Im_4^{\circ A}$) способами виникають питання щодо реалізації операцій ФК та визначення КСД, де зберігаються результати цих операцій, то це спонукатиме до повернення до попередніх: вербального ($\Im_{r=1}^{'A}$) та табличного ($\Im_{r=2}^{'A}$) описів алгоритму та їх цілеспрямованої верифікації щодо вказаних питань. Це **2-й етап верифікації** описів та коригування вербального та табличного з них:

$$\Re_2 \Leftrightarrow \bigcup_{r=1}^{2}\left(\Im_r^{'A} \xrightarrow{\Pi_r^2} \Im_r^{''A}\right) . \qquad (6)$$

Якщо при створенні опису алгоритму блок-схемним способом ($\Im_5^{\circ A}$) виникають питання щодо послідовності проведення операцій ФК, їхньої суті та визначення місць зберігання результатів цих операцій у КСД, то це спонукатиме до повернення до попередніх описів алго-

ритму ($\Im^{"A}_{r=1}$, $\Im^{"A}_{r=2}$, $\Im^{\circ A}_{r=3}$ і $\Im^{\circ A}_{r=4}$) та, у ході **3-го етапу верифікації** описів, їх коригування:

$$\Re_3 \Leftrightarrow \bigcup_{r=1}^{2}\left(\Im^{"A}_r \xrightarrow{\Pi^3_r} \Im^{'''A}_r\right) \bigcup_{r=3}^{4}\left(\Im^{\circ A}_r \xrightarrow{\Pi^1_r} \Im^{'A}_r\right). \quad (7)$$

Якщо виникають питання (сумніви) щодо правильності отриманих результатів під час функціонування програмного засобу, створеного за програмним описом $\Im^{\circ A}_6$, то це спонукатиме до повернення до цього опису для його коригування у відповідності до $\Im^{\circ A}_5$ або, може, і до всіх попередніх описів ($\Im^{'''A}_{r=1}$, $\Im^{'''A}_{r=2}$, $\Im^{'A}_{r=3}$, $\Im^{'A}_{r=4}$ та $\Im^{\circ A}_5$) алгоритму та, у ході **4-го етапу верифікації** описів, їх коригування.

$$\Re_4 \Leftrightarrow \bigcup_{r=1}^{2}\left(\Im^{'''A}_r \xrightarrow{\Pi^4_r} \Im^{''''A}_r\right) \bigcup_{r=3}^{4}\left(\Im^{'A}_r \xrightarrow{\Pi^2_r} \Im^{"A}_r\right) \bigcup$$

$$\bigcup\left(\Im^{\circ A}_5 \xrightarrow{\Pi^1_5} \Im^{'A}_5\right) \bigcup\left(\Im^{\circ A}_6 \xrightarrow{\Pi^1_6} \Im^{'A}_6\right). \quad (8)$$

Таким чином, процес верифікації $\Re$ всього ряду описів для всіх етапів $\rho$ можна представити як:

$$\Re \Leftrightarrow \bigcup_{\rho=1}^{4}\Re_\rho \Leftrightarrow$$

$$\Leftrightarrow \left\{\begin{array}{l} \Im^{\circ A}_1 \xrightarrow{\Pi^1_1} \Im^{'A}_1 \xrightarrow{\Pi^2_1} \Im^{"A}_1 \xrightarrow{\Pi^3_1} \Im^{'''A}_1 \xrightarrow{\Pi^4_1} \Im^{''''A}_1 \Leftrightarrow \Im^A_1 \\ \Im^{\circ A}_2 \xrightarrow{\Pi^1_2} \Im^{'A}_2 \xrightarrow{\Pi^2_2} \Im^{"A}_2 \xrightarrow{\Pi^3_2} \Im^{'''A}_2 \xrightarrow{\Pi^4_2} \Im^{''''A}_2 \Leftrightarrow \Im^A_2 \\ \Im^{\circ A}_3 \xrightarrow{\Pi^1_3} \Im^{'A}_3 \xrightarrow{\Pi^2_3} \Im^{"A}_3 \Leftrightarrow \Im^A_3 \\ \Im^{\circ A}_4 \xrightarrow{\Pi^1_4} \Im^{'A}_4 \xrightarrow{\Pi^2_4} \Im^{"A}_4 \Leftrightarrow \Im^A_4 \\ \Im^{\circ A}_5 \xrightarrow{\Pi^1_5} \Im^{'A}_5 \Leftrightarrow \Im^A_5 \\ \Im^{\circ A}_6 \xrightarrow{\Pi^1_6} \Im^{'A}_6 \Leftrightarrow \Im^A_6 \end{array}\right\}. \quad (9)$$

Блок-схема здійснення такої поетапної верифікації показана на рис. 1.

## 5. Екземпліфікація верифікації описів алгоритму

Екземпліфікація здійснюється на прикладі описів фрагменту реального алгоритму діючої підсистеми, представлених різними способами.

Приклад опису фрагменту алгоритму вербальним способом. ...«Якщо у ТБД БД К-1 "Параметри контролю реєстрації документів" значення реквізиту "Допустимий сумарний термін реєстрації документів у межах інтервалу контролю" менше сумарного терміну реєстрації документів то:

а) у поточному записі ТБД БД 6.1-4 "Терміни реєстрації документів" реквізиту "Порушення" надається зна-чення "2";

б) створюється "новий" запис таблиці ТБД БД 6.1-4 "Терміни реєстрації документів" (до "нового" запису з поточного здійснюється копіювання спільних даних) та для нього:

- підраховується і заноситься у новий запис ТБД БД 6.1-4 "Терміни реєстрації документів" реквізиту "Дата наступного контролю терміну реєстрації документів" сума значень цього реквізиту з старого запису та відповідного значення реквізиту "Термін перебування на обліку" ТБД БД К-1 "Параметри контролю реєстрації документів" (у даному разі значення підрахованого реквізиту становить дату зняття з обліку);

- у "новому" записі ТБД БД 6.1-4 "Терміни реєстрації документів" реквізиту "Порушення" надається значення "2";

в) записам ТБД БД 6.1-2 "Установчі дані", які мають єдине значення реквізиту "Вид документу" (визначається за значенням реквізитів "Ідентифікатор запису БД 6.1-2" та "Ідентифікатор "первинного" запису" з ТБД БД 6.1-1):

- реквізиту "Порушення" надається значення "2";

- реквізиту "Облік" надається значення "1" (тобто, потрібно взяти на облік);

- реквізиту "Дата встановлення на облік в БД 1.11" надається значення реквізиту "Дата наступного контролю терміну реєстрації документів" "старого" запису ТБД БД 6.1-4 "Терміни реєстрації документів" плюс 3 доби;

- реквізиту "Дата зняття з обліку" надається значення реквізиту "Дата наступного контролю терміну реєстрації докумен тів" "нового" запису ТБД БД 6.1-4 "Терміни реєстрації паспортних документів";

г) формується запис ТБД БД 1.11-1 "Відомості про осіб, які перевищили термін реєстрації документів" за відповідними даними ТБД БД 6.1-2 "Установчі дані" (формування запису у ТБД БД 1.11-1 реєструється за реквізитом "Дата, час уведення інформації до БД" цієї ТБД)».

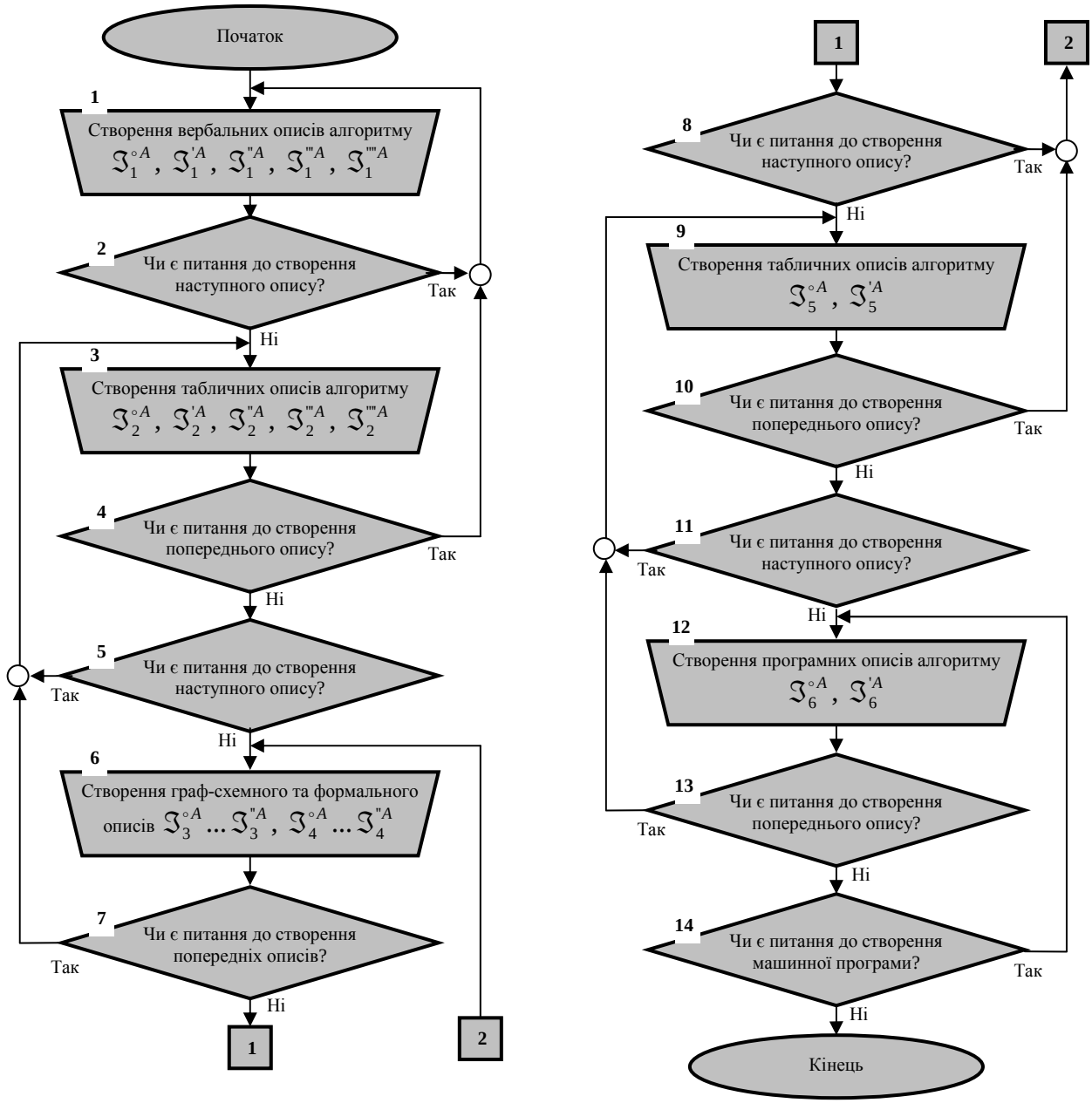


Рис. 1. Блок-схема здійснення поетапної верифікації описів алгоритмів

Приклад опису фрагмента алгоритму табличним способом приводиться в табл. 2 та 3

Таблиця 2. Приклад МТОА для процедури "А"

| **Процедури та умови, при яких вони відбуваються** | | **Результат процедури** (наслідки для таблиць БД, їх записів та реквізитів) | | | | |
|---|---|---|---|---|---|---|
| Процедури та загальні умови | Операції та умови їх здійснення | ТБД БД 6.1-4 "Терміни реєстрації документів" | | | | |
| | | запис ТБД | | "Дата наступного контролю терміну реєстрації документів" | "Порушення" | |
| | | поточний (старий) | новий | | старий запис | новий запис |
| **Процедура А** Обробка записів ТБД БД 6.1-4 "Терміни реєстрації документів" для контролю сумарного терміну реєстрації | **Операція А.1** визначення дати наступного контролю терміну реєстрації документів Е4 **Умова А.1** DST < ST | Не змінюється | Створюється шляхом копіювання спільних даних зі старого запи-су | Значення цього реквізиту старого запису ТБД БД 6.1-4 + значення реквізиту "Термін перебування на обліку" ТБД КПД-1 "Параметри контролю реєстрації документів" | 2 | 2 |

Таблиця 3. Приклад МТОА для процедури "Б"

| **Процедури та умови, при яких вони відбуваються** | | **Результат процедури** (наслідки для таблиць БД, їх записів та реквізитів) | | | | | |
|---|---|---|---|---|---|---|---|
| Процедури та загальні умови | Операції та умови їх здійснення | ТБД БД 6.1-2 "Установчі дані" | | | | ТБД БД 1.11-1 "Відомості про осіб, які перевищили термін реєстрації документів" | |
| | | "По-рушення" | "Об-лік" | "Дата взяття на облік в БД 1. 11" | "Дата зняття з обліку" | Запис таблиці | "Дата, час уведення інформації до БД" |
| **Процедура Б** Обробка записів ТБД БД 6.1-2 "Установчі дані" та БД 1.11-1 "Відомості про осіб, які перевищили термін реєстрації документів" для контролю сумарного терміну реєстрації | **Операція Б.1** Тотожна процедурі Б **Умова Б.1** DST < ST | " 2" | " 1" | Значення реквізиту "Дата наступного контролю терміну реєстрації документів" старого запису ТБД БД 6.1-4 + 3 доби | Значення реквізиту "Дата наступного контролю терміну реєстрації документів" нового запису ТБД БД 6.1-4 | Формується запис за відповідними даними ТБД БД 6.1-2 "Установчі дані" | Реєстрація |

Примітки: DST – значення реквізиту "Допустимий сумарний термін реєстрації документів у межах інтервалу контролю" ТБД БД К-1 "Параметри контролю реєстрації документів";
ST – значення реквізиту "Сумарний термін реєстрації документів одного виду" ТБД БД 6.1-4.

Приклад опису фрагмента алгоритму граф-схемним способом показано на рис. 2 та 3.

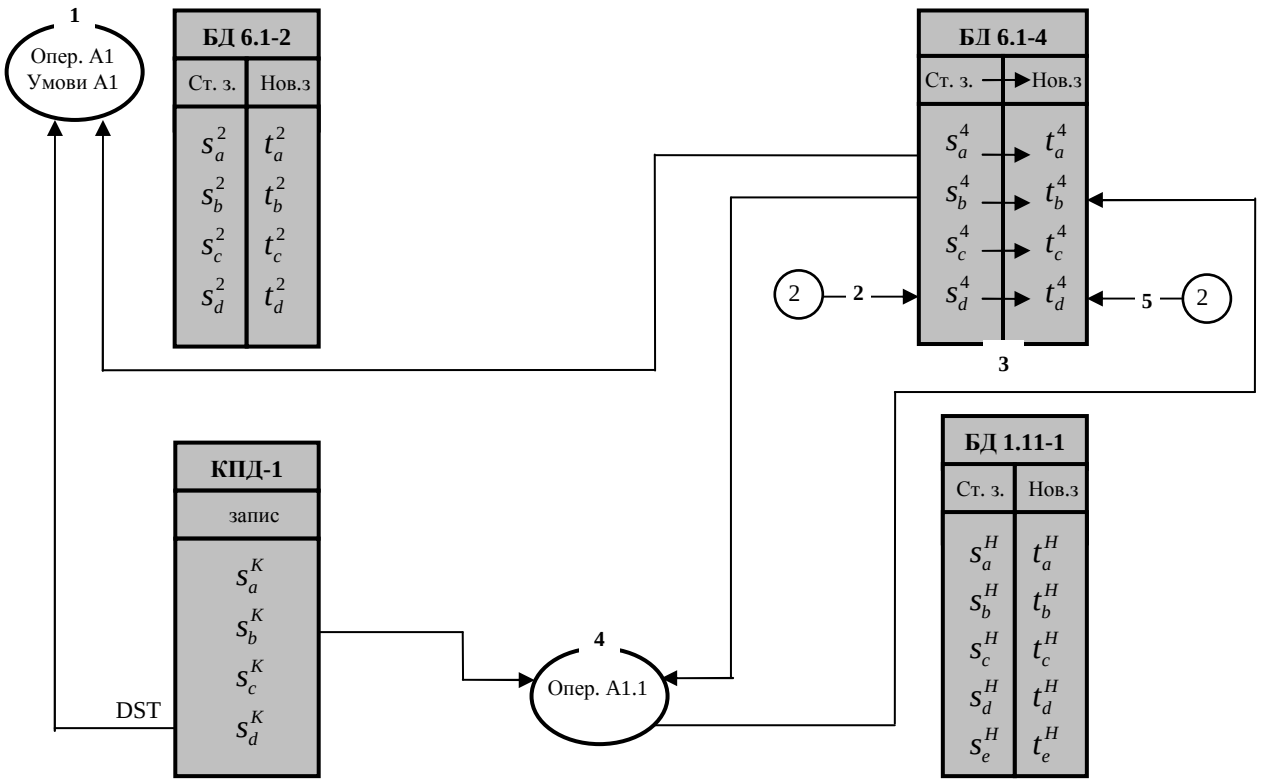


Рис. 2. Опис процедури "А" у відповідності до табл. 2

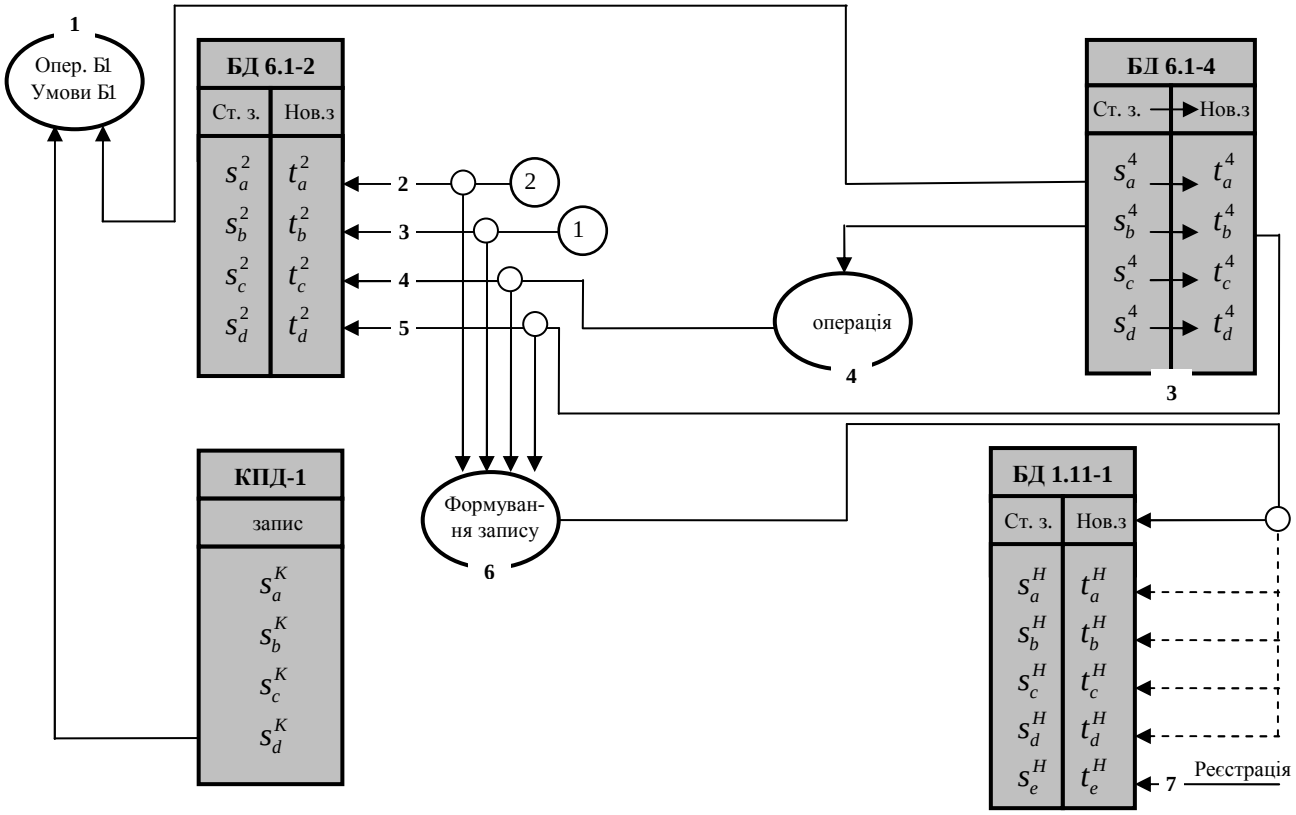


Рис. 3. Опис процедури "Б" у відповідності до табл. 3

Позначення, що прийняті на рис. 2 та 3, наведені у табл. 4 і 5.

Таблиця 4. Найменування реквізитів ТБД

| ТБД | Новий та старий запис реквізиту | |
|---|---|---|
| | **Код** | **Найменування** |
| БД 6.1-2 "Установчі дані" | $s_a^2$, $t_a^2$ | Порушення |
| | $s_b^2$, $t_b^2$ | Облік |
| | $s_c^2$, $t_c^2$ | Дата встановлення на облік в БД |
| | $s_d^2$, $t_d^2$ | Дата зняття з обліку |
| БД 6.1-4 "Терміни реєстрації документів" | $s_a^4$, $t_a^4$ | Сумарний термін реєстрації документів одного виду |
| | $s_b^4$, $t_b^4$ | Дата наступного контролю терміну реєстрації документів |
| | $s_c^4$, $t_c^4$ | Контроль |
| | $s_d^4$, $t_d^4$ | Порушення |
| БД К-1 "Параметри контролю реєстрації документів" | $s_a^K$ | Дата започаткування контролю |
| | $s_b^K$ | Термін перебування на обліку |
| | $s_c^K$ | Інтервал контролю |
| | $s_d^K$ | Допустимий сумарний термін реєстрації документів у межах інтервалу контролю |
| БД 1.11-1 "Відомості про осіб, які перевищили термін реєстрації документів" | $s_a^H$, $t_a^H$ | Порушення |
| | $s_b^H$, $t_b^H$ | Облік |
| | $s_c^H$, $t_c^H$ | Дата встановлення на облік в БД |
| | $s_d^H$, $t_d^H$ | Дата зняття з обліку |
| | $s_e^H$, $t_e^H$ | Дата та час уведення інформації до БД |

Таблиця 5. Найменування операцій

| Процедура | Операція | |
|---|---|---|
| | **Номер** | **Найменування** |
| Процедура "А" (див. табл. 2) | 1.1 | Перевірка умов А1 процедури А1 |
| | 1.2 | Надання значення "2" у старий запис реквізиту "Порушення" |
| | 1.3 | Створення нового запису шляхом копіювання спільних даних зі старого |
| | 1.4 | Розрахунок та занесення у новий запис "Дата наступного контролю терміну реєстрації документів" суму значень з старого запису та відповідного значення реквізиту "Термін перебування на обліку" БД К-1 |
| | 1.5 | Надання значення "2" у новий запис реквізиту "Порушення" |
| Процедура "Б" (див. табл. 3) | 2.1 | Перевірка умов Б1 процедури Б1 |
| | 2.2 | Надання значення "2" у новий запис реквізиту "Порушення" |
| | 2.3 | Надання значення "1" у новий запис реквізиту "Облік" |
| | 2.4 | Надання значення "Дата наступного контролю терміну реєстрації документів" старого запису БД 6.1-4 + 3 доби у новий запис "Дата взяття на облік в БД 1.11" БД 6.1-2 |
| | 2.5 | Надання значення "Дата наступного контролю терміну реєстрації документів" нового запису БД 6.1-4 у новий "Дата зняття з обліку" БД 6.1-2 |
| | 2.6 | Формування запису для БД 1.11-1 за відповідними даними БД 6.1-2 |
| | 2.7 | Здійснення реєстрації у р. "Дата, час уведення інформації до БД" |

Приклад опису фрагмента алгоритму формальним способом у вигляді математичних описів окремих операцій за процедурами наведено у табл. 6 та 7. Скорочення та визначення змінних наведені у табл. 4 та 5.

Таблиця 6. Формальний опис процедури "А"

| Математична операція | |
|---|---|
| **Номер** | **Код** |
| 1.1 | $(s_d^K < s_a^4) \Rightarrow \exists A1\big(f(A1) \neq \Theta\big)$, де $\Theta$ – пуста множина |
| 1.2 | $s_d^4 = 2$ |
| 1.3 | $t_a^4 = s_a^4, \quad t_b^4 = s_b^4,$ $t_c^4 = s_c^4, \quad t_d^4 = s_d^4$ |
| 1.4 | $t_b^4 = s_b^4 + s_b^K$ |
| 1.5 | $t_a^4 = 2$ |

Таблиця 7. Формальний опис процедури "Б"

| Математична операція | |
|---|---|
| **Номер** | **Код** |
| 2.1 | $(s_d^K < s_a^4) \Rightarrow \exists Б1\big(f(Б1) \neq \Theta\big)$ |
| 2.2 | $t_a^2 = 2$ |
| 2.3 | $t_b^2 = 1$ |
| 2.4 | $t_c^4 = s_b^4 + 3$ |
| 2.5 | $t_d^2 = t_b^4$ |
| 2.6 | $\{t_a^H, t_b^H, t_c^H, t_d^H\} = \{t_a^2, t_b^2, t_c^2, t_d^2\}$ |
| 2.7 | $t_e^H \neq \Theta$ |

Приклад опису фрагмента алгоритму блок-схемним способом показано на рис. 4.

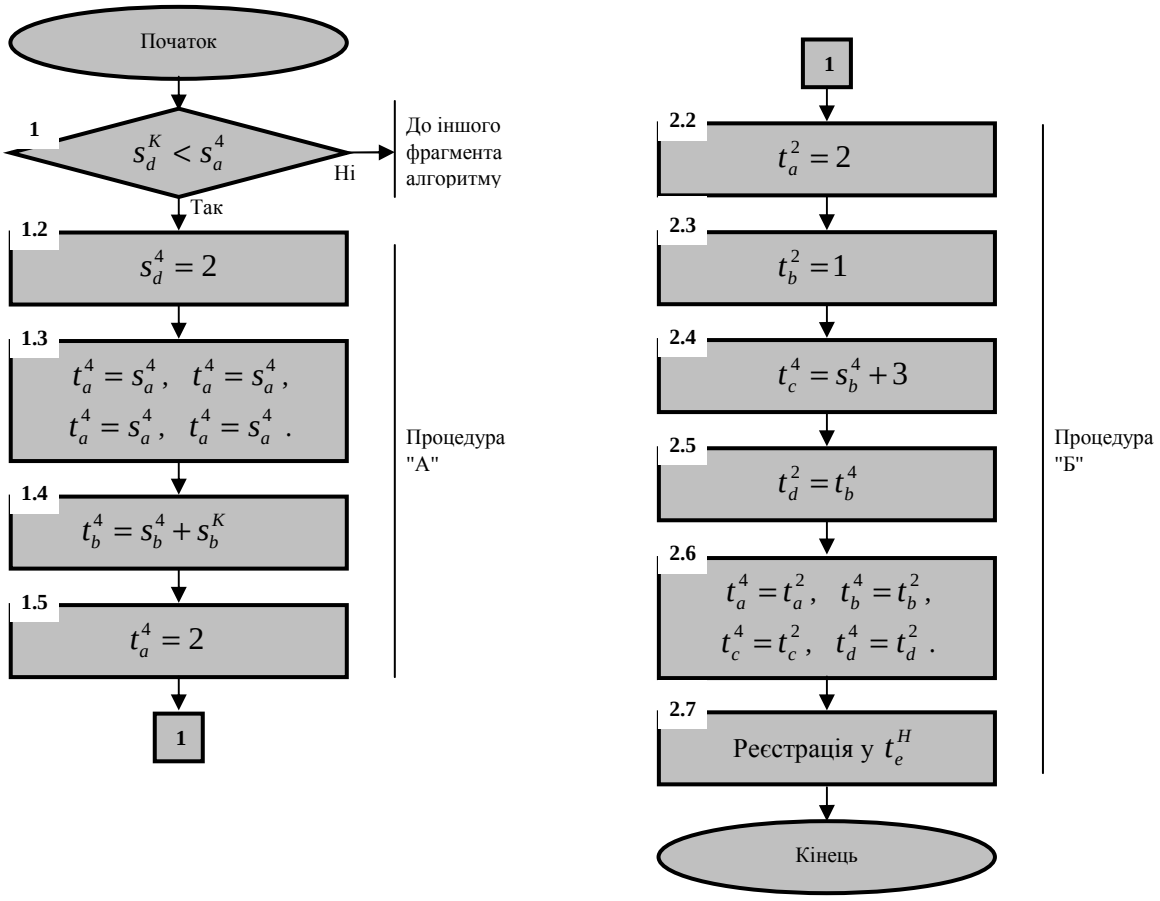


Рис. 4. Опис фрагмента алгоритму блок-схемним способом

## Висновки

У роботі запропоновано підхід до верифікації вербального опису алгоритму вирішення прикладної задачі у складі інформаційної системи. Такий опис створюється на початковій стадії проектування, під час визначення та збору вимог до функціонування такої задачі з боку користувача інформаційної системи та системного проектувальника. Практика показала, що якість такого опису далека від бажаної. Помилки через нечіткі або неоднозначні формулювання вимог можуть призвести до того, що виготовлений програмний продукт не буде задовольняти замовника – того самого користувача. Тому на етапах розробки проекту вимоги, а з ними і вербальний опис задачі, мають постійно уточнюватись як користувачем, так і системним проектувальником: користувачем – для уточнення самої задачі, а проектувальником – для коректного і якісного викладення її у вигляді відповідного опису. Для досягнення цього пропонується верифікувати початковий вербальний опис алгоритму за допомогою інших описів алгоритму, створених у табличний, граф-схемний, формальний та блок-схемний спосіб. Це дозволяє заздалегідь, до стадії програмування, поетапно виявити та виправити помилки та неточності як у вербальному, так і у інших описах, і створити досконалу блок-схему алгоритму для використання програмістами під час розробки програмного продукту.

Треба зазначити, що хоча наведені тут способи опису алгоритму вже давно відомі, але їх застосування у такому контексті запропоновано уперше. І ще, такий підхід можна розглядати як відгалуження методики проектування великих систем так званим методом формалізованих технічних завдань [12, с. 151], де замість проблемно-орієнтованих алгоритмічних мов використовується визначений ряд способів опису алгоритму.

Як перспективу подальшої роботи у цьому напрямку можна запропонувати застосування математичного апарату теорії графів для еквівалентного перетворення граф-схем з мінімізацією кількості вузлів, що може призвести до оптимізації структури КСД при реалізації заданого алгоритму.


1. *Каныгин Ю.М., Калитич Г.И.* "Основы теоретической информатики". – Киев: Наук. думка, 1990. – 232 с.
2. *Закон* України "Про Національну програму інформатизації" від 4.02.1998, № 74/98-ВР (із змінами, внесеними згідно із Законом від 13.09.2001, № 2684-III).
3. *ДСТУ* 2938-94. Системи оброблення інформації. Основні поняття. Терміни та визначення. Держстандарт України.
4. *Бабенко Л.П., Лавріщева К.М.* Основи програмної інженерії: Навч. посіб. – К.: Т-во "Знання", КОО, 2001. – 209 с.
5. *Software* Engineering: a European Respective. IEEE Computer Society Press, Los Almitos, California. – Washington, Brussels, Tokyo, 1993. – 682 p.
6. *ДСТУ* 2941-94. Системи оброблення інформації. Розроблення систем. Терміни та визначення. Держстандарт України.
7. *ДСТУ* 3918-99 (ISO/IEC 12207:1995) Інформаційні технології. Процеси життєвого циклу програмного забезпечення. Держстандарт України.
8. *Лавріщева К.М.* Програмна інженерія. – К., 2008. – 319 с.
9. *Андон Ф.И., Лаврищева Е.М., Суслов В.Ю. и др.* Основы инженерии качества программных систем. – 2-е изд., перераб. и доп. – Киев: Академпериодика, 2007. – 672 с.
10. *Вендров А.М.* Проектирование программного обеспечения экономических информационных систем: Учебник. – М.: Финансы и статистика, 2000. – 352 с.
11. *Колдовский В.* Разработка ПО: оценка результата. http://itc.ua/node/25631.
12. *Глушков В.М.* Основы безбумажной информатики. – М.: Наука. Гл. ред. Физ.-мат. лит., 1987. – 552 с.
13. *Алфёрова З.В.* Теория алгоритмов. – М.: Издательство "Статистика", 1973. – 164 с.
14. *Советский* энциклопедический словарь. – М.: Советская энциклопедия,1989. – 1632 с.

15. *ДСТУ* 2226-93. Автоматизовані системи. Терміни та визначення. Держстандарт України.
16. *ГОСТ* 19.701-90 Единая система программной документации. Схемы алгоритмов и программ.
17. *Башлыков А.А*. Проектирование систем принятия решений в энергетике. – М.: Энергоатомиздат, 1986. – 120 с.
18. *Калянов Г.Н.* CASE. Структурный системный анализ (автоматизация и применение). – М.: Лори, 1996. – 327 с.
19. *Энциклопедия* кибернетики / В 2-х томах. – Киев: Главная редакция Украинской советской энциклопедии АН УССР, 1974. – 1232 с.

Отримано 20.07.2009

***Про авторів:***

*Алексеєв Віктор Анатолійович*,
кандидат технічних наук,
завідуючий відділом,

*Терещенко Валерій Савелійович*,
кандидат технічних наук,
старший науковий співробітник.

***Місце роботи авторів:***

Інститут програмних систем
НАН України,
03187, Київ-187,
Проспект Академіка Глушкова, 40.
Тел.: (044) 526 4228; (044) 526 6191;
факс.: (044) 526 4228.
e-mail: alekseev@isofts.kiev.ua,
terek@isofts.kiev.ua