ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

INFORMATION TECHNOLOGY

УДК 65.011.56 **DOI: 10.20998/2079-0023.2021.01.13**

Є. П. ПАВЛЕНКО, В. М. БУТЕНКО, В. О. ГУБІН, С. В. ЛУБЕНЕЦЬ

ДОСЛІДЖЕННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ТИПІВ ДАНИХ ПРИ РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

У роботі розглядаються проблеми підвищення ефективності розробки програмного забезпечення, зокрема, питання зменшення термінів розробки програм та використання автоматизованого синтезу програм, що дозволить уникнути доопрацювання вихідного продукту. Програмне забезпечення потрібно протестувати разом з іншими системними компонентами у всіх поєднаннях, які можуть зустрітися. Тестування займає багато часу, бо приховані помилки виявляються при несподіваних взаємодіях програмних компонентів. При структурному аналізі діаграми потоків даних не є кінцевим результатом, це інструмент розробників. Спочатку будуються діаграми, а потім розробляються механізми, що забезпечують необхідну поведінку системи. Розвивається графічний підхід до вирішення проблеми автоматизації розробки програмних засобів, який базується на залученні візуальних форм представлення програм. Для будь-якого програмного об'єкта можна виділити кінцеве число станів, в яких він перебуває в кожен момент часу. Хід виконання програми тоді асоціюється з переходами об'єкта з одного стану в інший. Граф замінює текстову форму опису алгоритму програми, при цьому реалізується подання алгоритму в візуальній формі. Специфікація структур даних, а також установка міжмодульного інтерфейсу за даними відділена від опису структури алгоритму і елементів управління. Використовуються базові модулі і типи даних. Базові модулі є локальними обчислюваними функціями, на основі яких породжуються всі інші об'єкти технології. Типи даних описують синтаксичний і семантичний аспекти побудови даних, що використовуються в базових функціях. Розглянуто алгоритми пошуків маршрутів на орієнтованих графах. Під час визначення маршрутів з кореневої вершини в кінцеві використано властивості алгебри тризначної логіки. На підставі розглянутого підходу, а також з урахуванням його недоліків, був запропонований метод класифікації типів даних, заснований на реалізації часткового перебору маршрутів графу зв'язків програми та спосіб проектування ПЗ на його основі з урахуванням мінімізації часу і вартості проекту.

Ключові слова: програмне забезпечення, комп'ютерна інженерія, інформаційні системи, компоненти, частковий перебір маршрутів графу, витрати на розробку.

Е. П. ПАВЛЕНКО, В. М. БУТЕНКО, В. А. ГУБИН, С. В. ЛУБЕНЕЦ

ИССЛЕДОВАНИЕ МЕТОДОВ КЛАССИФИКАЦИИ ТИПОВ ДАННЫХ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНОЙ ИНЖЕНЕРИИ

В работе рассматриваются проблемы повышения эффективности разработки программного обеспечения, в частности, вопрос уменьшения сроков разработки программ и использования автоматизированного синтеза программ, что позволит избежать доработки исходного продукта. Программное обеспечение нужно протестировать вместе с другими системными компонентами во всех сочетаниях, которые могут встретиться. Тестирование занимает много времени, так как скрытые ошибки выявляются при неожиданных взаимодействиях программных компонентов. При структурном анализе диаграммы потоков данных не являются конечным результатом, это инструмент разработчиков. Сначала строятся диаграммы, а затем разрабатываются механизмы, обеспечивающие необходимое поведение системы. Развивается графический подход к решению проблемы автоматизации разработки программных средств, основанный на привлечении визуальных форм представления программ. Для любого программного объекта можно выделить конечное число состояний, в которых он находится в каждый момент времени. Граф заменяет текстовую форму описания алгоритма программы, при этом реализуется представление алгоритма в визуальной форме. Спецификация структур данных, а также установка межмодульных интерфейса по данным отделена от описания структуры алгоритма и элементов управления. Используются базовые модули и типы данных. Базовые модули являются локальными исчисляемыми функциями, на основе которых порождаются все остальные объекты технологии. Рассмотрены алгоритмы поиска маршрутов на ориентированных графах. На основании рассмотренного подхода, а также с учетом его недостатков, был предложен метод классификации типов данных, основанный на реализации частичного перебора маршрутов графа связей программы и способ проектирования ПО на его основе с учетом минимизации времени и стоимости проекта.

Ключевые слова: программное обеспечение, компьютерная инженерия, информационные системы, компоненты, частичный перебор маршрутов графа, затраты на разработку.

Y. P. PAVLENKO, V. M. BUTENKO, V. O. GUBIN, S. V. LUBENETS

RESEARCH OF DATA TYPE CLASSIFICATION METHODS WHEN DEVELOPING COMPUTER ENGINEERING SOFTWARE

The paper deals with the problems of increasing the efficiency of software development, in particular, the issue of reducing the time for developing programs and using automated synthesis of programs, which will avoid the revision of the original product. The software should be tested along with

other system components in all combinations that may occur. Testing is time-consuming because hidden bugs are revealed through unexpected interactions between software components. With structural analysis, data flow diagrams are not the end result, they are a developer tool. First, diagrams are built, and then mechanisms are developed to ensure the required system behavior. A graphical approach to solving the problem of automation of software development is being developed, based on the involvement of visual forms of program presentation. For any program object, you can select a finite number of states in which it is at each moment of time. The program progress is associated with the transition of an object from one state to another. The graph replaces the textual form of the description of the program algorithm, while the visual representation of the algorithm is realized. The specification of data structures, as well as the setting of intermodular interfaces according to data, is separated from the description of the structure of the algorithm and controls. Basic modules and data types are used. Basic modules are local calculable functions, on the basis of which all other technology objects are generated. Data types describe the syntactic and semantic aspects of constructing data used in base functions. Algorithms for finding routes on directed graphs are considered. When defining routes from the root vertex to the final ones, the properties of the algebra of three-valued logic were used. Based on the considered approach, as well as taking into account its shortcomings, a method for classifying data types was proposed, based on the implementation of a partial enumeration of the routes of the graph of program links and a method for designing software based on it, taking into account minimizing the time and cost of the project.

Keywords: software, computer engineering, information systems, components, partial enumeration of graph routes, development costs.

Вступ. Проблеми якості роботи програмного забезпечення (ПЗ) комп'ютерної інженерії зараз набули першорядну важливість. В різних джерелах докладно обговорюються фактори, що впливають на ефективність використання ПЗ на підприємствах, але слід проаналізувати головний фактор, що визначає якість використання ПЗ. Цей фактор – технологія створення і супроводу ПЗ.

Слід звернути увагу на фактор часу, який необхідно витратити на розробку ПЗ до введення його в експлуатацію, оскільки час розробки є основним критерієм, що визначає витрати на створення ПЗ. Вимоги до ПЗ, до його функцій можуть змінюватися, адже вони визначаються середовищем, в якому буде працювати ПЗ, отже час, як ресурс, має найбільш значну вагу при розрахунку економічної доцільності створення програмного продукту.

Для скорочення часу розробки і підвищення якості ПЗ розроблена достатня кількість технологій і підходів. Вони стосуються не тільки етапів життєвого циклу ПЗ, але й побудови структури компанії-розробника програмних продуктів.

Використання автоматичного синтезу програм дозволяє розробнику програмних продуктів реалізовувати систему графів переходів у вигляді програми, що дозволяє знизити вимоги до його кваліфікації, автоматизувати процес програмування. Результат розробки мовами високого рівня залежить від використовуваної мови програмування, необхідно визначити класи, атрибути, області дії змінних.

Постановка проблеми. Якість ПЗ визначається методами і способами моделювання предметної області, які дозволяють здійснювати розробку структур баз даних, інформаційної структури розроблюваного ПЗ.

Програма повинна бути спроектована таким чином, щоб використовувати заздалегідь обумовлену кількість ресурсів – обсягу пам'яті, часу процесора тощо. ПЗ потрібно протестувати разом з іншими системними компонентами у всіх поєднаннях, які можуть зустрітися. Тестування займає багато часу, бо приховані помилки виявляються при несподіваних взаємодіях програмних компонентів.

Складність ПЗ збільшилася, виділилися області, в яких помилки або недостатня якість програм можуть завдати шкоди, яка перевищить економічний ефект від їх використання.

Ефективність розробки ПЗ визначається сукупністю багатьох факторів. Існують різноманітні підходи до оцінки якості розробки ПЗ, які мають переваги і недоліки. Таким чином, виникає задача дослідження технологій використання автоматизованого синтезу програм, які дозволяють скоротити час розробки програмного продукту, а також дослідження підвищення ефективності їх застосування.

Аналіз стану проблеми. Досить часто плани створення складних програмних засобів і БД готуються і оцінюються некваліфіковано, на основі думок замовників і розробників про необхідні функції та можливі якості ПЗ. Тому помилки при визначенні необхідних показників якості, оцінки трудомісткості, вартості та тривалості створення ПЗ є поширеним явищем. Часто проекти ПЗ не відповідають вимогам до характеристик якості, не вкладаються в графіки і бюджет розробки.

Застосування стандартів, що регламентують процеси життєвого циклу ПЗ, може служити основою для забезпечення якості програмних засобів, однак потрібне коригування та адаптація деяких положень стандартів стосовно особливостям технологій програмування.

Виділяються два типи підходів до розробки ПЗ – структурний та об'єктно-орієнтований. Оскільки в об'єктно-орієнтованому ПЗ екземпляри класів обмінюються повідомленнями, для кожного класу визначають повідомлення, які надходять його об'єктам і на їх основі будують діаграми переходу [1]. Також будують моделі станів для кожного об'єкта і визначають списки подій, що змінюють стан об'єктів [2].

Після виділення класів і їх опису можна побудувати моделі процесів, які повинні бути реалізовані в майбутній програмі [3].

Якщо користувач правильно вказав всі характеристики даного, потрібно побудувати екземпляр об'єкта і включити його в множину даних [4].

Структурний аналіз є альтернативою процесам та етапам об'єктно-орієнтованого аналізу. Після проведення структурного аналізу будується модель системи, описана діаграмами потоків даних й іншими елементами. Ці діаграми дають формальну модель проблеми. Потім можна приступити до визначення класів і об'єктів різними способами.

Один із способів полягає в тому, що спочатку формується словник даних і потім виконується аналіз контекстних діаграм моделі [5].

Інший спосіб полягає в аналізі окремих діаграм потоків даних. В якості об'єктів розглядаються зовнішні сутності, сховища даних, керуючі перетворення, сховища керуючих сутностей. В якості класів – потоки даних, потоки управління. Перетворення даних розглядається як операція над існуючими об'єктами або як поведінка об'єкту, який створюється для виконання необхідного перетворення [6].

В структурному аналізі вхідні та вихідні дані вивчаються, поки не досягнуть вищого рівня абстракції. Процес перетворення вхідних даних у вихідні є основним. Після визначення основної сутності в діаграмі потоків даних вивчається вся інфраструктура, простежуються вхідні та вихідні потоки даних від неї, групуються процеси і стани, як основні, так і побічні.

При структурному аналізі діаграми потоків даних не є кінцевим результатом, це інструмент розробників. Спочатку будуються діаграми, а потім розробляються механізми, що забезпечують необхідну поведінку системи. Зберігається тільки продукт структурного аналізу високого рівня абстракції. Він є незалежним від проекту системи [7].

Засоби автоматизації програмування орієнтовані на наближення мови програмування до способу мислення програміста, в бездирективному методі організації розробки програм. Підвищення продуктивності праці програміста пов'язується з більшою зрозумілістю програмного коду.

При дедуктивному синтезі програм використовується формальний метод побудови програм. Розробка програми з заданої специфікації розглядається як задача доказу існування потрібного рішення.

Індуктивний синтез програм ґрунтується на знаходженні спільних для заданої множини прикладів процедур, що ведуть до розв'язання задачі. Індуктивний синтез програм може здійснюватися на синтаксичному рівні, коли використовуються зовнішні ознаки програм, або на семантичному рівні, коли в основі лежить семантика моделі програми [8].

Розвивається також графічний підхід до вирішення проблеми автоматизації розробки програмних засобів, який базується на залученні візуальних форм представлення програм. Застосування графічних методів дозволяє підвищити продуктивність праці програміста. Графічна форма запису забезпечує високий рівень їх структуризації, дотримання технологічної культури програмування, пропонує надійний стиль програмування [9].

Розробка програмного забезпечення для потреб залізниці має специфіку, зокрема, необхідність підтримувати розподілені обчислення в різних компонентах інформаційних систем на залізничному транспорті [10], [11]. Вибір технології розробки програмного забезпечення для таких систем пропонується виконувати з урахуванням засобів, описаних в [12].

Метод вирішення проблеми. Для подання алгоритмів оберемо модель програмного модуля з дискретними станами. Для будь-якого програмного об'єкта можна виділити кінцеве число станів, в яких він перебуває в кожен момент часу. Хід виконання програми тоді асоціюється з переходами об'єкта з одного стану в інший.

Програма інтерпретується деякою математичною функцією

$$M: in(D) \to out(D);$$

де $in(D)$ – множина вхідних даних програмного модуля M,

$out(D)$ – множина вихідних даних програмного модуля M.

Граф станів G визначається як орієнтований граф, вершини якого відповідають стану, а дуги – переходам системи зі стану в стан. Кожна вершина графа позначається відповідною функцією f.

Дуги графа інтерпретуються як події. Будемо розглядати подію, як зміну стану програмного модулю, яка впливає на виконання програми. Відповідна подія визначає подальший хід виконання програми, яка побудована згідно з певним алгоритмом.

Активізація деякої події залежить від стану об'єкта, яке визначається даними D модулю M.

Розглянемо множину предикативних функцій $P = \{P_1, P_2, \dots, P_n\}$. Це функції, які в залежності від значень даних D дорівнюють 0 або 1. Дугам графа G поставимо у відповідність функції P. Подія, яка виконує перехід на графі, відбувається, якщо модель об'єкта на даному кроці роботи алгоритму знаходиться в стані S_i та відповідна предикативна функція $P_{ij}(D)$ дорівнює 1.

В якості алгоритмічної моделі методу використаємо множину $< D, F, P, G >$, де D – структури даних, F – множина функцій предметної області, P – множина предикативних функцій, що обробляють структури даних предметної області, G – граф станів об'єкту.

Граф замінює текстову форму опису алгоритму програми, при цьому реалізується подання алгоритму в візуальній формі та відбувається декомпозиція основних компонент опису алгоритму програми. Специфікація структур даних, а також установка міжмодульного інтерфейсу за даними відділена від опису структури алгоритму і елементів управління.

Запропонована алгоритмічна модель визначає деяку функцію і може служити вихідними даними для побудови алгоритмічних моделей інших програм. Таким чином, цей метод допускає побудову ієрархічних алгоритмічних моделей. Структура алгоритмічної моделі залежить від способу декомпозиції об'єкта програмування на множини станів і подій, які визначаються предикативними функціями.

В даній технології використовують базові модулі і типи даних. Базові модулі є локальними обчислюваними функціями, на основі яких породжуються всі інші об'єкти технології. Типи даних описують синтаксичний і семантичний аспекти побудови даних, що використовуються в базових функціях, а також і в об'єктах.

Модуль є замкнутою програмною одиницею з набором входів і покажчиків виходів, яку можна

викликати з будь-якого іншого модуля програми і окремо компілювати. В модулі виділяють його зовнішність (інтерфейс) – частину, за допомогою якої модуль пов'язується із зовнішнім середовищем – іншими модулями, операційною системою.

Тип функції визначається як множина відображень з області визначення функції в область результату. Областю визначення функції є декартове множення значень кількох типів формальних параметрів, областю результатів – множина значень деякого одного типу даних.

Функцію розглядають як опис абстрактного типу: $T_1, T_2, \dots, T_n \to T_r$, де $T_1, T_2, \dots, T_n$ – типи формальних параметрів функції, T_r – тип результату функції.

Вихідні дані та результати обчислень базових модулів розміщуються в списку типів даних функції, тому тип базового модуля визначається як відображення типів даних з області визначення на область їх значень.

В технології об'єктно-орієнтованого програмування в якості програмних одиниць розглядаються об'єкти. Вони мають конкретний зміст сенс і діють в рамках предметної області програмування. В предметній області вже визначений термінологічний словник даних. Для кожної предметної області будується інформаційне середовище, що дозволяє уніфікувати проектування програмних модулів різними розробниками.

Проблема передачі інформації від однієї програми до іншої представляє собою одну з проблем, яка служить джерелом найбільшої кількості помилок в програмному забезпеченні. Вирішення цієї проблеми досягається за умови введення обмежень на способи і методи побудови міжмодульного інформаційного інтерфейсу, а також за рахунок розробки засобів автоматизації побудови таких інтерфейсів.

Інформація про поділ даних за ознакою їх використання в об'єктах необхідна при побудові на основі об'єкта виконуваного exe-модуля, для виконання операції інкапсуляції об'єктів та при тестуванні об'єктів.

Операція агрегації з сукупності наявних об'єктів породжує новий об'єкт-агрегат. Проблема автоматизації породження базових модулів представляє практичний інтерес. Як вирішення цієї проблеми пропонується використовувати операцію інкапсуляції об'єктів-агрегатів.

В об'єкті-агрегаті віднесення того чи іншого даного до певного класу залежить від маршруту роботи алгоритму на керуючому графові об'єкта. Задача класифікації даних для об'єктів-агрегатів пов'язана з вирішенням проблеми виділення всіх незалежних маршрутів, які можуть бути реалізовані в графі агрегату.

Задача побудови всіх незалежних маршрутів з кореневої вершини в кінцеві вершини графа еквівалентна задачі декомпозиції вихідного графа на сукупність частин графа, таких, що $G = \cup R_i$, де R_i – орієнтовані маршрути з кореневої вершини в кінцеві.

Кожен з маршрутів є лінійним графом (в загальному випадку вони можуть містити і цикли), а розвиток обчислювального процесу на графі відбувається по одному з виділених напрямів.

Класифікаційні ознаки даних визначаються розташуванням об'єктів на маршруті обчислень. Якщо кілька модулів на маршруті мають спільний параметр, то його класифікаційна ознака буде визначатися типом його першого входження. Наприклад, для маршруту $R_1: M_1 \to M_2 \to M_3 \to M_4$ об'єкт O_1 зустрічається в двох модулях: M_2 та M_3, в модулі M_2 цей об'єкт обчислюється і не вимагає попередньої ініціалізації, а в модулі M_3 об'єкт ініціюється і повинен прийняти значення заздалегідь. У зв'язку з тим, що модуль M_2 в цьому маршруті зустрічається раніше, то O_1 не потрібно заздалегідь ініціювати. Таким чином, на цьому маршруті об'єкт O_1 необхідно віднести до обчислюваних. Якщо об'єкт зустрічається вперше і одразу ініціюється, то його слід віднести до зовнішніх даних.

Використаємо алгоритми пошуків маршрутів на орієнтованих графах і запропонуємо метод класифікації типів даних, заснований на реалізації часткового перебору маршрутів графу зв'язків. Під час визначення маршрутів з кореневої вершини в кінцеві використаємо властивості алгебри тризначної логіки. Маршрути орієнтованого графа будемо описувати списками вершин. З точки зору класифікації даних агрегату схеми маршрутів визначимо диз'юнкти алгебри тризначної логіки.

Пропонується такий алгоритм вирішення задачі часткового перебору маршрутів графу.

Крок 1. Вершини графа кодуються символами впорядкованої множини будь-яким способом, але так, щоб коренева вершина мала найменше значення, а кінцеві – найбільше.

Крок 2. Для поточного стану маршруту R_i шукається шлях переходу з останньої вершини в будь-яку вільну вершину.

Крок 3. Якщо перехід можливий, то до маршруту додається один символ. Якщо перехід неможливий, то розглядається наступна вершина списку вільних вершин і так до тих пір, поки не скінчиться список вершин.

Крок 4. Якщо перехід з поточної вершини в будь-яку з вільних вершин неможливий, то відбувається повернення за схемою маршруту на один символ назад.

Крок 5. При досягненні кінцевої вершини алгоритм повертається на один символ назад.

Крок 6. Алгоритм завершує свою роботу, якщо список вільних вершин кореневої вершини вичерпаний.

Ефективність алгоритму залежить від зв'язності графа. Для повнозв'язаного графа реалізується повний перегляд усіх схем маршрутів. В інших випадках число перевірок реалізованості маршрутів на графі дорівнює сумі потужностей множин досяжності і недосяжності.

Залежність розкиду усередненої складності запропонованого алгоритму в залежності від ступеня

графа отримана експериментально, в результаті моделювання графів різної структури. З ростом кількості вершин графу складність алгоритму збільшується також для лінійних графів, і раціональним порядком графа можна вважати 7–8.

Застосування досліджуваного алгоритму та моделей може бути ефективним при синтезі спеціалізованого програмного забезпечення транспортного призначення. Серед класу зазначеного ПЗ є такі компоненти, які використовуються у підсистемах транспорту для управління складними технологічними процесами критичними до безпеки їх функціонування. Однак, для зазначеного застосування моделі слід дослідити особливості функціонування та експлуатації відповідного обладнання та визначити ймовірність впливу помилок програмного забезпечення на роботу такого спеціалізованого обладнання.

Майбутнім удосконаленням моделі є включення до її складу компонентів, які б оптимізували не тільки відповідні множини досяжності і недосяжності, але й більш високі властивості алгоритму спеціального призначення. Це дало б змогу більш ефективніше реалізовувати тестування алгоритмів та програм спеціалізованих комп'ютерних систем, зокрема транспортного призначення. Але таке удосконалення потребує суттєвих витрат часу й буде проводитись в подальшому.

Висновки. В роботі проведено дослідження технологій автоматизованого синтезу програм, був проведений аналіз характеристик якості програмного забезпечення інформаційних систем.

На підставі проведеного дослідження розроблено алгоритм рішення задачі оптимізації графу програми, вдосконалено метод класифікації типів даних, заснований на реалізації часткового перебору маршрутів графу. Отримані результати роботи випробувані на реальному прикладі – при моделювання графів програм різної структури.

Розроблений узагальнений алгоритм в результаті невеликих змін може бути використаний для часткового перебору маршрутів графів програм при проектуванні різних видів ПЗ, і розроблюване програмне забезпечення може бути вдосконалено за рахунок збільшення його функціональності та скорочення витрат на розробку.

Дослідження відкриває перспективи подальших розробок та удосконалень спеціалізованого програмного забезпечення комп'ютерних систем транспортного призначення критичних до безпеки.

Список літератури

1. Патрикеев Ю. *Объектно-ориентированное проектирование*. URL: http://www.object.newmail.ru/oop1.html (дата звернення: 05.06.2020).
2. Буч Г. *Объектно-ориентированный анализ и проектирование с примерами приложений*. URL: https://www.twirpx.com/file/279137 (дата звернення: 05.06.2020).
3. *Проектирование программного обеспечения*. URL: http://window.edu.ru/library/pdf2txt/965/28965 (дата звернення: 27.06.2020).
4. Маклафлин Б., Поллайс Г., Уэст Д. *Объектно-ориентированный анализ и проектирование.* URL: https://library.bntu.by/maklaflin-brett-obektno-orientirovannyy-analiz-i-proektirovanie (дата звернення: 27.06.2020).
5. McMenamin S., Palmer J. *Essential Systems Analysis*. New York, 1984. New York: Yourdon Press. 267 p.
6. Ward P., Mellor S. *Structured Development for Real-time Systems*. *Vol.1*. New York: Pearson Technology Group, 2008. 176 p.
7. *Объектно-ориентированный анализ и проектирование*. URL: http://www.hardline.ru/1/5/1390/1789-6.htm (дата звернення: 27.06.2020).
8. Новиков Ф.А. *Системы представления знаний*. URL: http://window.edu.ru/catalog/pdf2txt/372/60372 (дата звернення: 27.06.2020).
9. Вельбицкий И.В. *Технология программирования.* URL: https://www.twirpx.com/file/170607 (дата звернення: 27.06.2020).
10. Listrovoy S. V., Butenko V. M., Bryksin V. O., Golovko O. V. Development of method of definition maximum clique in a non-oriented graph. *EasternEuropean Journal of Enterprise Technologies*. 2017. Vol. 5, № 4 (89). P. 12–17. EID: 2-s2.0-85032585697 DOI: 10.15587/1729-4061.2017.111056).
11. Лістровий С. В., Панченко С. В., Мойсеєнко В. І., Бутенко В. М. *Математичне моделювання в розподілених інформаційно-керуючих системах залізничного транспорту.* Харків: ФОП Бровін О. В, 2017. 220 с.
12. Павленко Е. П., Бутенко В. М., Губин В. А. Исследование методов разработки программного обеспечения компьютерной инженерии на основе типовых программных элементов. *Вестник Нац. техн. ун-та «ХПИ»: сб. науч. тр. Темат. вып.: Системный анализ, управление и информационные технологии*. Харьков: НТУ «ХПИ». 2019. № 1. С. 67–71.

References (transliterated)

1. Patrikeev Y. *Ob"yektno-oriyentirovannoye proyektirovaniye* [Object Oriented Design] Available at: http://www.object.newmail.ru/oop1.html (accessed 05.06.2020).
2. Booch G. *Ob"yektno-oriyentirovannyy analiz i proyektirovaniye s primerami prilozheniy* [Object-oriented analysis and design with examples of applications] Available at: https://www.twirpx.com/file/279137 (accessed 05.06.2020).
3. *Proyektirovaniye programmnogo obespecheniya* [Software design] Available at: http://window.edu.ru/library/pdf2txt/965/28965 (accessed 27.06.2020).
4. McLaughlin B., Pollays G., West D. *Ob"yektno-oriyentirovannyy analiz i proyektirovaniye* [Object Oriented Analysis and Design] Available at: https://library.bntu.by/maklaflin-brett-obektno-orientirovannyy-analiz-i-proektirovanie (accessed 27.06.2020).
5. McMenamin S., Palmer J. *Essential Systems Analysis*. New York, 1984. NY, Yourdon Press Publ. 267 p.
6. Ward P., Mellor S. *Structured Development for Real-time Systems*. *Vol.1*. New York: Pearson Technology Group Publ., 2008. 176 p.
7. *Ob"yektno-oriyentirovannyy analiz i proyektirovaniye* [Object-oriented analysis and design] Available at: http://www.hardline.ru/1/5/1390/1789-6.htm (accessed 27.06.2020).
8. Novykov F. *Sistemy predstavleniya znaniy* [Knowledge Presentation Systems] Available at: http://window.edu.ru/catalog/pdf2txt/372/60372 (accessed 27.06.2020).
9. Wielbitsky Y. *Tekhnologiya programmirovaniya* [Programming technology] Available at: https://www.twirpx.com/file/170607 (accessed 27.06.2020).
10. Listrovoy S. V., Butenko V. M., Bryksin V. O., Golovko O. V. Development of method of definition maximum clique in a non-oriented graph. *EasternEuropean Journal of Enterprise Technologies*. 2017, vol. 5, no. 4 (89), p. 12–17. EID: 2-s2.0-85032585697 DOI: 10.15587/1729-4061.2017.111056).
11. Listrovoy S. V., Panchenko S. V., Mojseenko V. I., Butenko V. M. *Matematychne modelyuvannya v rozpodilenykh informatsiyno-keruyuchykh systemakh zaliznychnoho transportu* [Mathematical modeling in distributed information-control systems of railway transport]. Kharkiv, 2017. 220 p.
12. Pavlenko Y. P., Butenko V. M., Gubin V. O. *Issledovaniye metodov razrabotki programmnogo obespecheniya komp'yuternoy inzhenerii*

na osnove tipovykh programmnykh elementov [Research of methods for developing computer engineering software based on standard software elements] *Vestnik Nats. tekhn. un-ta «KHPI»: sb. nauch. tr. Temat. vyp.: Sistemnyy analiz, upravleniye i informatsionnyye tekhnologii* [Bulletin of NTU "KhPI". Series: Systems Analysis, Management and Information Technology]. Kharkov, 2019, no. 1, pp. 67–71.

Надійшла (received) 08.02.2021

Відомості про авторів / Сведения об авторах / About the Authors

Павленко Євген Петрович – кандидат технічних наук, доцент, Український державний університет залізничного транспорту, доцент кафедри спеціалізованих комп'ютерних систем; м. Харків, Україна; ORCID: http://orcid.org/0000-0002-7626-9933; e-mail: evgenijpavlenko821@gmail.com

Бутенко Володимир Михайлович – кандидат технічних наук, доцент, Український державний університет залізничного транспорту, доцент кафедри спеціалізованих комп'ютерних систем; м. Харків, Україна; ORCID: http://orcid.org/0000-0001-9958-3960; e-mail: butenko@kart.edu.ua

Губін Вадим Олександрович – Національний університет радіоелектроніки, старший викладач кафедри штучного інтелекту; м. Харків, Україна; ORCID: http://orcid.org/0000-0003-1850-1930; e-mail: vadim.gubin@nure.ua

Лубенець Сергій Васильович – кандидат технічних наук, доцент, Харківський національний університет імені В. Н. Каразіна, доцент кафедри міжнародних відносин, міжнародної інформації та безпеки; м. Харків, Україна; ORCID: http://orcid.org/0000-0003-1061-8763; e-mail: S.Lubenec@karazin.ua

Павленко Евгений Петрович – кандидат технических наук, доцент, Украинский государственный университет железнодорожного транспорта, доцент кафедры специализированных компьютерных систем; г. Харьков, Украина; ORCID: http://orcid.org/0000-0002-7626-9933; e-mail: evgenijpavlenko821@gmail.com

Бутенко Владимир Михайлович – кандидат технических наук, доцент, Украинский государственный университет железнодорожного транспорта, доцент кафедры специализированных компьютерных систем; г. Харьков, Украина; ORCID: http://orcid.org/0000-0001-9958-3960; e-mail: butenko@kart.edu.ua

Губин Вадим Александрович – Национальный университет радиоэлектроники, старший преподаватель кафедры искусственного интеллекта; г. Харьков, Украина; ORCID: http://orcid.org/0000-0003-1850-1930; e-mail: vadim.gubin@nure.ua

Лубенец Сергей Васильевич – кандидат технических наук, доцент, Харьковский национальный университет имени В. Н. Каразина, доцент кафедры международных отношений, международной информации и безопасности; г. Харьков, Украина; ORCID: http://orcid.org/0000-0003-1061-8763; e-mail: S.Lubenec@karazin.ua

Pavlenko Yevhen Petrovych – Candidate of Technical Sciences (Ph. D.), Docent, Ukrainian State University of Railway Transport, Associate Professor at the Department of the Specialized Computer Systems; Kharkiv, Ukraine; ORCID: http://orcid.org/0000-0002-7626-9933; e-mail: evgenijpavlenko821@gmail.com

Butenko Vladimir Mihajlovych – Candidate of Technical Sciences (Ph. D.), Docent, Ukrainian State University of Railway Transport, Associate Professor at the Department of the Specialized Computer Systems; Kharkiv, Ukraine; ORCID: http://orcid.org/0000-0001-9958-3960; e-mail: butenko@kart.edu.ua

Gubin Vadim Oleksandrovych – National University of Radio Electronics, senior lecturer at the Department of Artificial Intelligence; Kharkiv, Ukraine; ORCID: http://orcid.org/0000-0003-1850-1930; e-mail: vadim.gubin@nure.ua

Lubenets Serhii Vasylovych – Candidate of Technical Sciences (Ph. D.), Docent, V. N. Karazin Kharkiv National University, Associate Professor of International Relations, International Information and Security; Kharkiv, Ukraine; ORCID: http://orcid.org/0000-0003-1061-8763; e-mail: S.Lubenec@karazin.ua