БІОЛОГІЧНІ ТА МЕДИЧНІ ПРИЛАДИ І СИСТЕМИ

УДК 681.5.017:616-71

О. М. Роїк[1], А. В. Поплавський[1], А. П. Ткачук[1], Д. П. Присяжний[1]

МЕТОДИ ЗАМІЩЕННЯ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ ПРИЙНЯТТЯ РІШЕНЬ В СИСТЕМАХ ДІАГНОСТИКИ СКЛАДНИХ ОБ'ЄКТІВ

[1]Вінницький національний технічний університет

Анотація: Моделі складних об'єктів (як медичних так і технічних) можна розглядати як багатополюсні досліджувані кола. При цьому, в технічних об'єктах полюсами являються провідники друкованих плат між якими знаходяться досліджувані компоненти, параметри яких вимірюються. А в медичних об'єктах полюсам відповідають так звані меридіани, кожен з яких характеризується репрезентативними біологічними точками акупунктури (БАТ). Досліджуваними тут є параметри між точками акупунктури, для вимірювання яких застосовуються методи Фоля і Накатані. Однак ці методи не враховують об'єктивно існуючих взаємозв'язків компонентів в досліджуваних об'єктах.

В останній час широкий розвиток отримали методи вимірювань із штучним розчленуванням замкнених кіл. Ці методи полягать у реконфігурації складних об'єктів, за допомогою деякого комутатора, у коло типу трикутник, в якому одна з його гілок є досліджуваним компонентом, який шунтується двома іншими компоне нтами, які утворюються під час декомпозиції об'єктів діагнстування (ОД) шляхом об'єднання з несполученими полюсами досліджуваного компонента в один вузол.

На другому етапі здійснюється саме вимірювання параметрів досліджуваного компонента за допомогою порівняння з параметрами деякого зразкового елемента деяким пристроєм врівноваження (ПВ).

В статті також, розроблено методи заміщення для підвищення точності прийняття рішень під час вимірювань у системах діагностки параметрів елементарних компонент складних об'єктів.

Ключові слова: об'єкт діагностики, параметри елементарних компонент, пристрій врівноваження, вимірювальний перетворювач.

Аннотация: Модели сложных объектов (как медицинских так и технических) можно рассматривать как многополюсные исследуемые круга. При этом в технических объектах полюсами являются проводники печатных плат между которыми находятся исследуемые компоненты, параметры которых измеряются. А в медицинских объектах полюсам соответствуют так называемые меридианы, каждый из которых характеризуется репрезентативными биологическими точками акупунктуры (БАТ). Исследуемыми здесь есть параметры между точками акупунктуры, для измерения которых применяются методы Фолля и Накатани. Однако эти методы не учитывают объективно существующих взаимосвязей компонентов в исследуемых объектах.

В последнее время широкое развитие получили методы измерений с искусственным расчленением замкнутых кругов. Эти методы заключается в реконфигурации сложных объектов, с помощью некоторого коммутатора, в круг типа треугольник, в котором одна из его ветвей является исследуемым компонентом, который шунтируется двумя другими компонентами, которые образуются во время декомпозиции объектов диагностирования (ОД) путем об соединение с несоединенными полюсами исследуемого компонента в один узел.

На втором этапе осуществляется именно измерения параметров исследуемого компонента посредством сравнения с параметрами некоторого образцового элемента некоторым устройством уравновешивания.

В статье также, разработаны методы замещения для точности принятия решений при измерениях в системах диагносткы параметров элементарных компонент сложных объектов.

Ключевые слова: объект диагностирования, параметры элементарних компонент, устройство уравновешивания, измерительный преобразователь.

Abstract: Models of complex objects (both medical and technical) can be considered as multipolar studied circles. In this case, in the technical objects, the poles are conductors of printed circuit boards between which the investigated components are located, parameters of which are measured. And in medical facilities, the poles correspond to the so-called meridians, each of which is characterized by representative biological points of acupuncture. The investigated here are the parameters between the points of acupuncture, which are used to measure the methods of Volle and Nakatani. However, these methods do not take into account the objectively existing interconnections of components in the objects being studied.

Recently, the methods of measurements with artificial dismemberment of closed circles have been widely developed. These methods consist of reconfiguring complex objects, using a certain switch, in the circle of a triangle type, in which one of its branches is an explored component that is shunted by two other components that are formed during the decomposition of the objects of diagnosis (OD) by means of The connection with the united poles of the studied component into one node. At the second stage, it is precisely measuring the parameters of the investigated component by comparing with some parameters of some exemplary element by some balancing device.

In the article also, substitution methods have been developed to increase the accuracy of decision making during measurements in the diagnostic systems of the parameters of the elementary components of complex objects.

Key words: Parameters of elementary components, equilibration device, measuring transducer.

DOI: https://doi.org/10.31649/1999-9941-2018-42-2-4-9.

Вступ

Моделі складних об'єктів (як медичних так і технічних) можна розглядати як багатополюсні досліджувані кола. При цьому, в технічних об'єктах полюсами являються провідники друкованих плат між якими знаходяться досліджувані компоненти, параметри яких вимірюються. А в медичних об'єктах полюсам відповідають так звані меридіани, кожен з яких характеризується репрезентативними біологічними точками акупунктури (БАТ). Досліджуваними тут є параметри між точками акупунктури, для

вимірювання яких застосовуються методи Фоля і Накатані. Однак ці методи не враховують об'єктивно існуючих взаємозв'язків компонентів в досліджуваних об'єктах.

Актуальність

В останній час широкий розвиток отримали методи вимірювань із штучним розчленуванням замкнених кіл [1-5]. Ці методи полягать у реконфігурації складних об'єктів, у коло типу трикутник, в якому одна з його гілок є досліджуваним компонентом $\dot{Y}_x$, який шунтується двома іншими компонентами $\dot{Y}_s$ і $\dot{Y}_h$, які утворюються під час декомпозиції об'єктів діагнстування (ОД) шляхом об'єднання з несполученими полюсами досліджуваного компонента в один вузол. На другому етапі здійснюється саме вимірювання параметрів досліджуваного компонента $\dot{Y}_x$ за допомогою порівняння з параметрами деякого зразкового елемента Y_o деяким пристроєм врівноваження (ПВ). На рис. 1 наведена узагальнена структурна схема вимірювального перктворювача (ВП) параметрів досліджуваного компонента $\dot{Y}_x$, де $\boldsymbol{G_{ПК}}$ - досліджуване коло пасивних компонентґ.

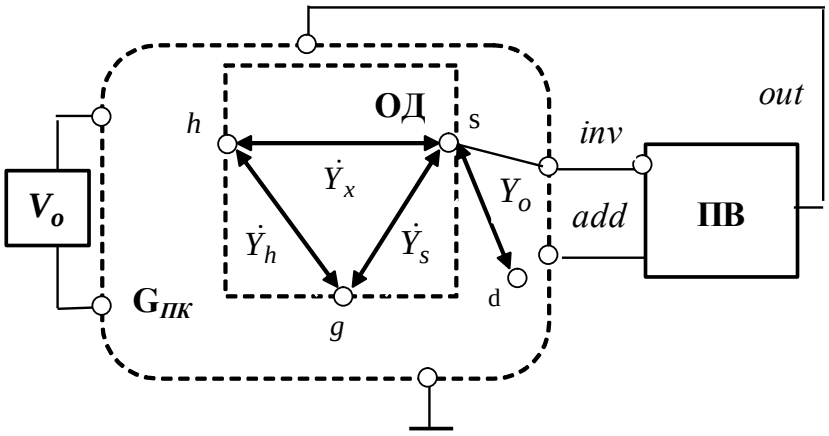
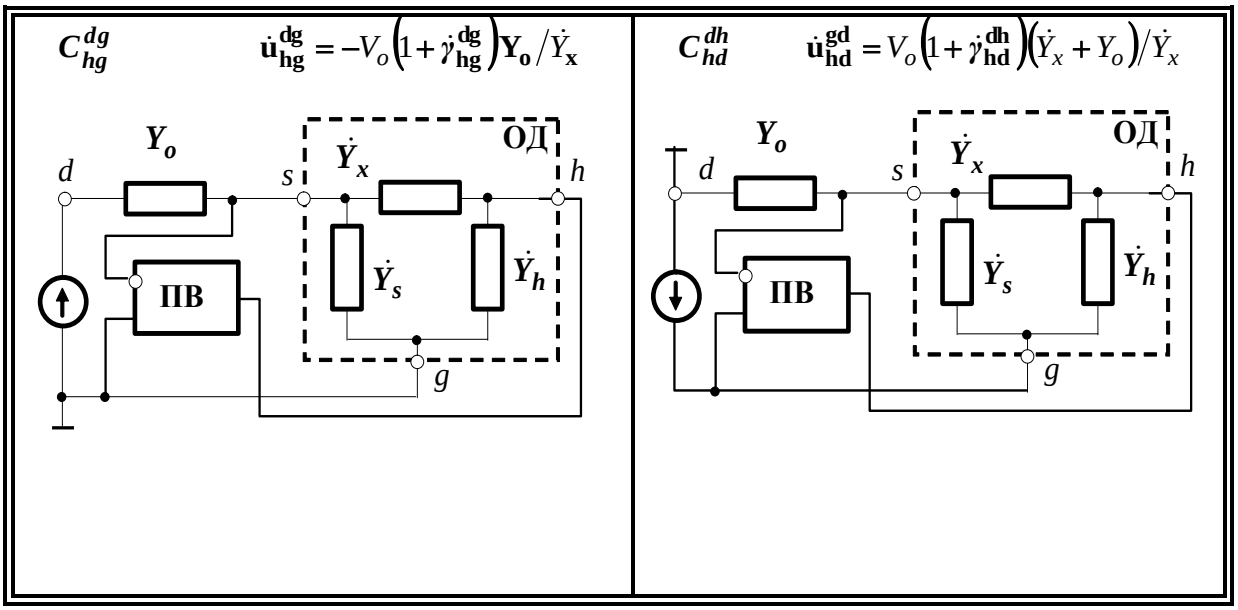


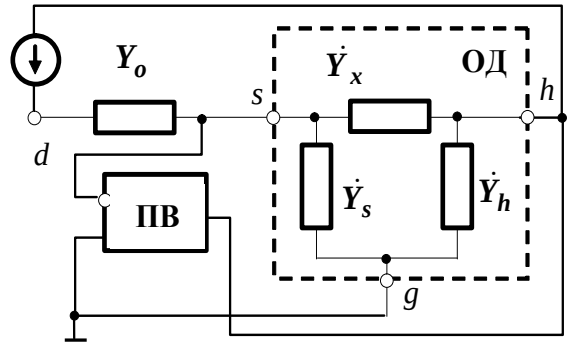
Рисунок 1 – Узагальнена структурна схема вимірювального перктворювача (ВП)

В роботах [1-5] розроблено комплекс базових структур (див. табл. 1), що реалізують узагальнену структуру ВП

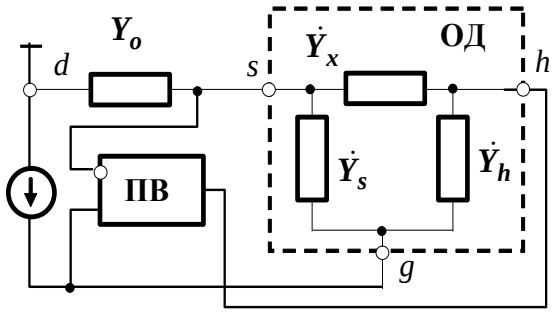
Таблиця 1 – Комплекс базових структурних схем вимірювальних перетворювачів



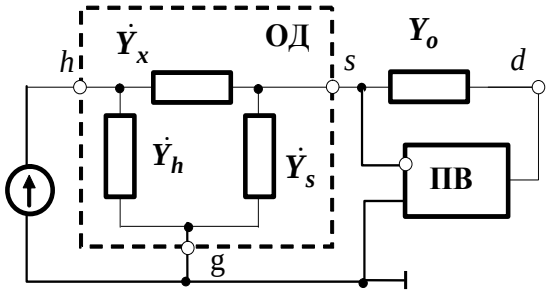
C_{hg}^{dh} $\dot{u}_{hg}^{dh} = -V_o\left(1+\dot{\gamma}_{hg}^{dh}\right)Y_o/\left(\dot{Y}_x + Y_o\right)$



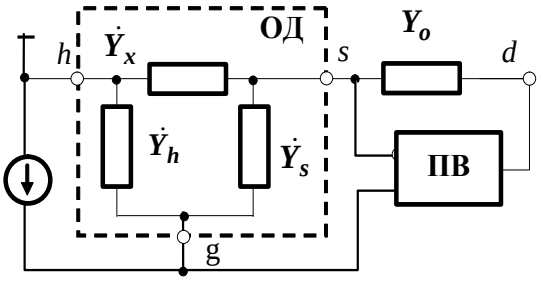
C_{hd}^{gh} $\dot{u}_{hd}^{gh} = V_o\left(1+\dot{\gamma}_{hg}^{dh}\right)\left(\dot{Y}_x + Y_o\right)/Y_o$



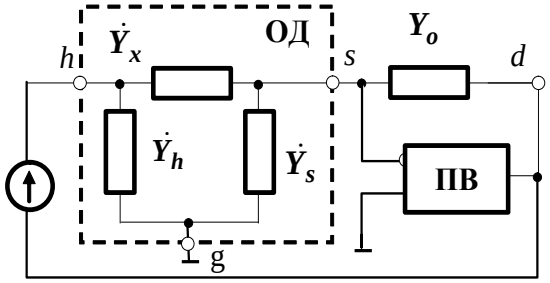
C_{dg}^{hg} $\dot{u}_{dg}^{hg} = -V_o\left(1+\dot{\gamma}_{dg}^{hg}\right)\dot{Y}_x/Y_o$



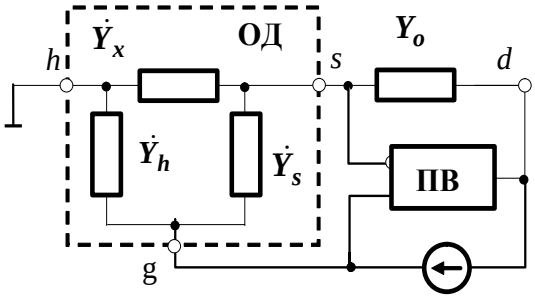
C_{dh}^{gh} $\dot{u}_{dh}^{gh} = V_o\left(1+\dot{\gamma}_{dh}^{gh}\right)\left(\dot{Y}_x + Y_o\right)/Y_o$



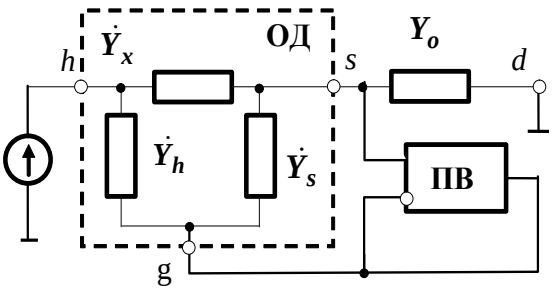
C_{dg}^{hd} $\dot{u}_{dg}^{hd} = -V_o\left(1+\dot{\gamma}_{dg}^{hd}\right)\dot{Y}_x/\left(\dot{Y}_x + Y_o\right)$



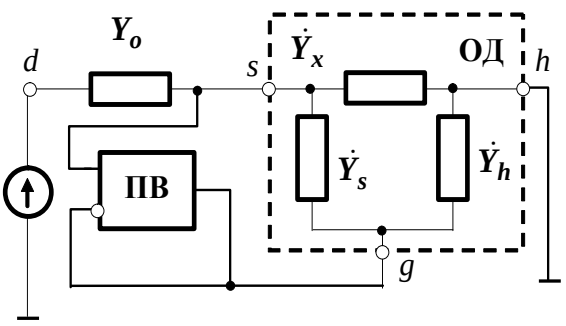
C_{dh}^{gd} $\dot{u}_{dh}^{gd} = V_o\left(1+\dot{\gamma}_{dh}^{gd}\right)\left(\dot{Y}_x + Y_o\right)/\dot{Y}_x$

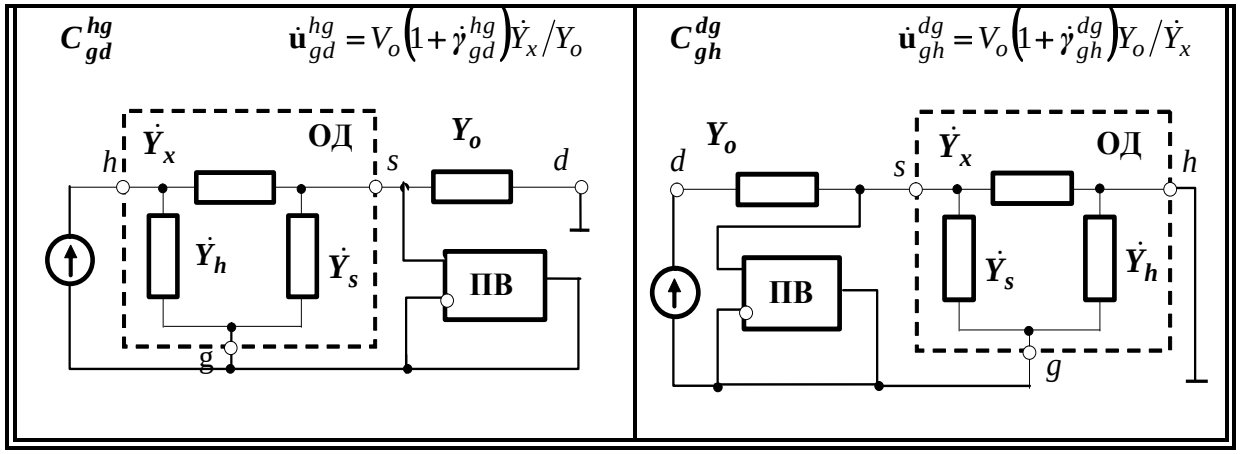


C_{gd}^{hd} $\dot{u}_{gd}^{ho} = V_o\left(1+\dot{\gamma}_{gd}^{ho}\right)\dot{Y}_x/\left(\dot{Y}_x + Y_o\right)$



C_{gh}^{dh} $\dot{u}_{gh}^{dh} = V_o\left(1+\dot{\gamma}_{gh}^{dh}\right)Y_o/\left(\dot{Y}_x + Y_o\right)$





Стан рівноваги базових структур описується матричним рівнянням:

$$\dot{U} = V_o\,\dot{W}_o\left(1+\dot{\gamma}\right), \tag{1}$$

де V_o – сигнал тестового впливу, $\dot{U}$ - діагональна матриця шуканих параметрів досліджуваних компонент $\dot{Y}_x$, $\dot{W}_o$ – діагональна матриця, в ідеальному випадку, співвідношень параметрів кола пасивних компонентг $G_{ПК}$, $\left(1+\dot{\gamma}\right)$ – вектор-стовпець, що визначає мультиплікативну похибку вимірювань.

При цьому $\dot{\gamma}$ можна визначити за виразом:

$$\dot{\gamma} = -\left[1+\alpha\,\dot{\beta}\right]^{-1},$$

де α - крутизна перетворень ПВ, $\dot{\beta}$ - нормалізуючі множники (коефіцієнти зворотного зв'язку) ПВ.

У свою чергу, нормалізуючі множники визначаються співвідношеннями параметрів досліджуваного кола пасивних компонент $G_{ПК}$ (див. табл.2):

$$\dot{\beta} = \dot{Y}_{чис}/\left(\dot{Y}_x+\dot{Y}_s+Y_o\right),$$

де $\dot{Y}_{чис}$ - визначається параметрами чисельника нормалізуючого множника відповідної базової структури.

Таблиця 2 – Значення елементів матриці рівняння врівноваження (1)

№	$\dot{U} = V_o\,\dot{W}_o\left(1+\dot{\gamma}_{(\cdot)(\cdot)}^{(\cdot)(\cdot)}\right)$				$\dot{\gamma}_{(\cdot)(\cdot)}^{(\cdot)(\cdot)} = -\left[1+\alpha\dot{\beta}_{(\cdot)(\cdot)}^{(\cdot)(\cdot)}\right]^{-1}$
	φ_h	φ_d	φ_g	$\dot{W}_o$	$\dot{\beta}$
1	$\dot{u}_{hg}^{dg}$	V_o	0	$\dot{w}_{hg}^{dg} = -Y_o/\dot{Y}_x$	$\dot{\beta}_{hg}^{dg} = \dot{Y}_x/\left(\dot{Y}_x+\dot{Y}_s+Y_o\right)$
2	$\dot{u}_{hg}^{dh}$	$\dot{u}_{hg}^{dh}+V_o$	0	$\dot{w}_{hg}^{dh} = -Y_o/\left(Y_o+\dot{Y}_x\right)$	$\dot{\beta}_{hg}^{dh} = \left(\dot{Y}_x+Y_o\right)/\left(\dot{Y}_x+\dot{Y}_s+Y_o\right)$
3	$\dot{u}_{hd}^{gd}$	0	V_o	$\dot{w}_{hd}^{gd} = \left(\dot{Y}_x+Y_o\right)/\dot{Y}_x$	$\dot{\beta}_{hd}^{gd} = \dot{Y}_x/\left(\dot{Y}_x+\dot{Y}_s+Y_o\right)$
4	$\dot{u}_{hd}^{gh}$	0	$\dot{u}_{hd}^{gh}+V_o$	$\dot{w}_{hd}^{gh} = \left(\dot{Y}_x+Y_o\right)/Y_o$	$\dot{\beta}_{hd}^{gh} = Y_o/\left(\dot{Y}_x+\dot{Y}_s+Y_o\right)$
5	V_o	$\dot{u}_{dg}^{hg}$	0	$\dot{w}_{dg}^{hg} = -\dot{Y}_x/Y_o$	$\dot{\beta}_{dg}^{hg} = Y_o/\left(\dot{Y}_x+\dot{Y}_s+Y_o\right)$

6	$\dot{u}_{dg}^{hd}+V_o$	$\dot{u}_{dg}^{hd}$	**0**	$\dot{w}_{dg}^{hd}=-\dot{Y}_x/(\dot{Y}_x+Y_o)$	$\dot{\beta}_{dg}^{hd}=(\dot{Y}_x+Y_o)/(\dot{Y}_x+\dot{Y}_s+Y_o)$
7	**0**	$\dot{u}_{dh}^{gh}$	V_o	$\dot{w}_{dh}^{gh}=(\dot{Y}_x+Y_o)/Y_o$	$\dot{\beta}_{dh}^{gh}=Y_o/(\dot{Y}_x+\dot{Y}_s+Y_o)$
8	**0**	$\dot{u}_{dh}^{gd}$	$\dot{u}_{dh}^{gd}+V_o$	$\dot{w}_{dh}^{gd}=(\dot{Y}_x+Y_o)/\dot{Y}_x$	$\dot{\beta}_{dh}^{gd}=\dot{Y}_x/(\dot{Y}_x+\dot{Y}_s+Y_o)$
9	**0**	V_o	$\dot{u}_{gh}^{dh}$	$\dot{w}_{gh}^{dh}=Y_o/(\dot{Y}_x+Y_o)$	$\dot{\beta}_{gh}^{dh}=(\dot{Y}_x+Y_o)/(\dot{Y}_x+\dot{Y}_s+Y_o)$
10	**0**	$\dot{u}_{gh}^{dg}+V_o$	$\dot{u}_{gh}^{dg}$	$\dot{w}_{gh}^{dg}=Y_o/\dot{Y}_x$	$\dot{\beta}_{gh}^{dg}=\dot{Y}_x/(\dot{Y}_x+\dot{Y}_s+Y_o)$
11	V_o	**0**	$\dot{u}_{gd}^{hd}$	$\dot{w}_{gd}^{hd}=\dot{Y}_x/(\dot{Y}_x+Y_o)$	$\dot{\beta}_{gd}^{hd}=(\dot{Y}_x+Y_o)/(\dot{Y}_x+\dot{Y}_s+Y_o)$
12	$\dot{u}_{gd}^{hg}+V_o$	**0**	$\dot{u}_{gd}^{hg}$	$\dot{w}_{gd}^{hg}=\dot{Y}_x/Y_o$	$\dot{\beta}_{gd}^{hg}=Y_o/(\dot{Y}_x+\dot{Y}_s+Y_o)$

Систему індексації елементів у таблицях 1 і 2 можна розглядати як алгоритм побудови конкретних структурних схем. А саме, верхні індекси висзначають полюси досліджуваного кола пасивних компонент $G_{ПК}$, між якими підключається джерело сигналу тестового впливу. Лівий нижній індекс визначає полюс подачі сиганлу врівноваження відносно вузла кола $G_{ПК}$, що визначається правим нижнім індексом.

Мета

Метою роботи є підвищення точності вимірювань методами заміщення за рахунок усунення мультиплікативної складової похибок.

Розв'язання задачі

Для розв'язання задачі пропонується у вимірювальний переиворювачь ввести додатковий зразковий елемент Y_o, як це показано для узагальненої структури на рис.2.

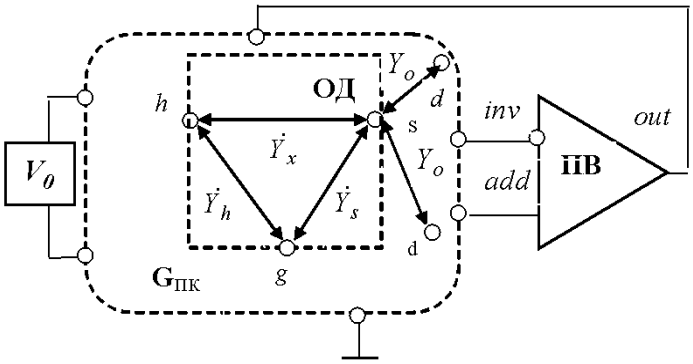


Рисунок 2 – Узагальнена структурна схема удосконаленого вимірювального перктворювача

При цьому процес вимірювання здійснюється у два етапи. На першому з них вимірюються параметри досліджуваного компонента $\dot{Y}_x$, підключаючи додатковий зразковий елемент Y_o до вузла g, У результаті отримаємо функцію перетворення:

$$\dot{U}=V_o\dot{W}_x\left(1+\dot{\gamma}_{(\cdot)(\cdot)}^{(\cdot)(\cdot)}\right).$$

На другому, вимірюються параметри додаткового зразкового елемента Y_o, підключаючи досліджуваний компонент $\dot{Y}_x$ також до вузла g, Нове перетворення буде визначатися виразом:

$$\dot{U}=V_o\dot{W}_o\left(1+\dot{\gamma}_{(\cdot)(\cdot)}^{(\cdot)(\cdot)}\right).$$

Відповідні переключення здійснюються деяким комутатором (на рис. 2 не показаний). Таким чи-

ном ми забезпечуємо в обох вимірювальних перетврюваннях буде забезпечуватись однакове шунтування входів прстрою врівноваження ПВ.

При цьому, очевидно, що в обох випадках забезпечується однакове значення мультиплікативної складової похибки $\dot{\gamma}$, а значення нормалізуючого множника буде визначатися виразом:

$$\dot{\beta} = \dot{Y}_{чис} / \left(\dot{Y}_x + \dot{Y}_s + 2Y_o \right).$$

Очеведно, що якщо розділити результати обох цих перетворень:

$$\dot{U} = W_x / W_o ,$$

то кінцевий результат буде вільний від мультиплікативної похибки вимірювань, а також і від значення сигналу тестового впливу, що значно спрощує його реалізацію.

Висновки

1. Розглянуто системний підхід до розв'язання задачі штучного розчленування замкнених кіл під час вимірювання параметрів елементів в системах діагностики складних об'єктів.
2. Проаналізовано мультиплікативні похибки вимірювань параметрів елементів у замкнених колах.
3. Запропоновано методи заміщення для підвищення точності прийняття рішень під час діагностики складних об'єктів.

Перелік використаних джерел

[1] О. М. Роїк, *Контроль і діагностика радіоелектронної апаратури на етапах її виробництва*. Вінниця, Україна: УНІВЕСУМ, 2000.

[2] О. М. Роїк, І. Р. Арсенюк, та В. І. Месюра, *Перетворення параметрів елементів замкнених кіл*. Вінниця, Україна: УНІВЕРСУМ-Вінниця, 2004.

[3] О. М. Роїк, та І. Р. Арсенюк *Діагностування аналогових пристроїв радіоелектронної апаратури*. Вінниця, Україна: УНИВЕРСУМ–Вінниця, 2005.

[4] О. М. Роїк, *Інваріантні перетворення параметрів елементів складних об'єктів*. Вінниця, Україна: УНІВЕСУМ-Вінниця, 2001.

[5] О. М. Роїк, та С. А. Яремко *Методи і засоби моделювання телемедичних систем функціонального стану людини*. Вінниця, Україна: ВНТУ, 2012.

Стаття надійшла: 28.08.18.

Відомості про авторів

Роїк Олександр Митрофанович – д.т.н., професор, завідувач кафедри менеджменту та безпеки інформаційних систем, Вінницький національний технічний університет, Хмельницьке шосе, 95, м. Вінниця.

Поплавський Анатолій Вацлавович – к.т.н., доцент кафедри менеджменту та безпеки інформаційних систем, Вінницький національний технічний університет, Хмельницьке шосе, 95, м. Вінниця.

Ткачук Анастасія Павлівна – студентка групи 1ПІ-15б факультету ІТКІ, Вінницький національний технічний університет, Хмельницьке шосе, 95, м. Вінниця.

Присяжний Дмитро Петрович – інженер Центру інформаційних технологій і захисту інформації, Вінницький національний технічний університет, Хмельницьке шосе, 95, м. Вінниця.

O. M. Roik[1], A. V. Poplavskiy[1], A. P. Tkachuk[1], D. P. Prysyazhnyy[1]

SUBSTITUTION METHODS HAVE TO INCREASE THE ACCURACY OF DECISION IN THE DIAGNOSTIC SYSTEMS OF COMPLEX OBJECTS

[1]Vinnytsia National Technical University

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.622

О. Д. Азаров[1], О. І. Черняк[1], В. В. Залізецький[1]

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ОПРАЦЮВАННЯ ДАНИХ ДИСТАНЦІЙНО-РОЗПОДІЛЕНИХ СИСТЕМ ТА ПОШУКУ ОБ'ЄКТІВ НА МІСЦЕВОСТІ

[1]Вінницький національний технічний університет

Анотація. В умовах євроінтеграції, подорожі за кордон стали більш доступними для українців і актуальною стає потреба добре орієнтуватись на незнайомій місцевості. Звісно зараз є безліч застосувань, що поліпшують життя туристам, наприклад безкоштовні Google карти вбудовані в більшість сучасних смартфонів на платформі Android. Але інколи потрібна інформація про розташування більше ніж 20-ти найближчих об'єктів, що не дозволяють зробити Google карти. Окрім того сейсморозвідка вимагає розміщення сенсорів на певній відстані один від одного з покриттям певної території. Потрібні засоби для контролю процесу розміщення сенсорів та відображення відсотку покриття території з врахуванням радіусу дії сенсорів. Також останнім часом, безпілотні літальні апарати активно завойовують своє місце в різних сферах життя. Потрібні інструменти для зручного керування такими апаратами. У статті пропонується програмне забезпечення для опрацювання даних дистанційно-розподілених систем та пошуку об'єктів на місцевості, що може використовуватись для вирішення вищеописаних задач. Запропоноване рішення розроблено з використанням сучасного стеку технологій, є надійним, незалежним від платформи, відносно дешевим та оснащене власним API.

Ключові слова: GPS, Google API, Web-сервіс, API, геолокація, сейсморозвідка, дистанційно-розподілені системи, пошук об'єктів на місцевості.

Аннотация. Благодаря евроинтеграции, путешествия за границу стали более доступными для украинцев и актуальной становится потребность хорошо ориентироваться на незнакомой местности. Конечно сейчас есть множество приложений, улучшающих жизнь туристам, например, бесплатные Google карты встроены в большинство современных смартфонов на платформе Android. Но иногда нужна информация о расположении более 20-ти ближайших объектов, что не позволяют сделать Google карты. Кроме того, сейсморазведка требует размещения сенсоров на определенном расстоянии друг от друга с покрытием определенной территории. Нужны средства для контроля процесса размещения сенсоров и отображения процента покрытия территории с учетом радиуса действия сенсоров. Также в последнее время, беспилотные летательные аппараты активно завоевывают свое место в различных сферах жизни. Необходимы инструменты для удобного управления такими аппаратами. В статье предлагается программное обеспечение для обработки данных дистанционно-распределенных систем и поиска объектов на местности, что может использоваться для решения вышеописанных задач. Предложенное решение разработано с использованием современного стека технологий, является надежным, независимым от платформы, относительно дешевым и оснащено собственным API.

Ключевые слова: GPS, Google API, Web-сервис, API, геолокация, сейсморазведка, дистанционно-распределенные системы, поиск объектов на местности.

Abstract. Thanks to the European integration, traveling abroad became more accessible for Ukrainians and the need to navigate the unfamiliar terrain is becoming urgent. Of course now there are many applications that improve the lives of tourists, for example, free Google maps are built into most modern smartphones on the Android platform. But sometimes you need information about the location of more than 20 nearby objects that do not allow you to make Google maps. In addition, seismic prospecting requires the placement of sensors at a certain distance from each other with the coverage of a certain area. We need tools to control the placement of sensors and display the percentage coverage of the territory, taking into account the range of sensors. Also recently, unmanned aerial vehicles are actively gaining their place in various spheres of life. Tools are needed for convenient control of such devices. The article proposes software for processing data from remote-distributed systems and searching for objects on the ground, which can be used to solve the above-described tasks. The proposed solution is developed using a modern stack of technologies, it is reliable, platform independent, relatively cheap and equipped with its own API.

Keywords: GPS, Google API, Web-service, API, geolocation, seismic exploration, remotely-distributed systems, objects locating.
DOI: https://doi.org/10.31649/1999-9941-2018-42-2-10-15.

Вступ

В умовах євроінтеграції, подорожі за кордон стали більш доступними для українців і актуальною стає потреба добре орієнтуватись на місцевості та швидко дізнатись де знаходиться найближчий банк, кафе, стоматолог чи автостоянка. Звісно на сьогодні є безліч застосувань, що поліпшують життя. Є різні навігатори, смартфони із вбудованими GPS датчиками та Google чи іншими картами. Усе це створює можливості для комфортного подорожування [1]. Проте інколи виникають специфічні потреби, наприклад отримати інформацію про розташування усіх об'єктів певного типу в межах певної місцевості, для якоїсь звітності, аналізу чи статистики. Google карти не дозволяють вивантажувати та відображати такі масиви інформації. Та це й не потрібно для такого типу застосунку.

Окрім того сейсморозвідка вимагає розміщення сенсорів на певній відстані один від одного з покриттям певної території [2]. При цьому було б дуже зручно бачити в реальному часі скільки території покрито, де розміщено сенсори та де потрібно встановити наступний сенсор.

Також останнім часом безпілотні літальні апарати (БПЛА), які завжди були надбанням військових,

завойовують своє місце ще й у різних сферах цивільного життя [3]. Щодня людство знаходить їм нові застосування, тому потрібне програмне забезпечення (ПЗ) для вирішення цих задач.

Аналіз останніх наукових досліджень і публікацій

Теоретичним обґрунтуванням процесів геопозиціонування займався Липкін І. А. [4]. Питання розробки систем моніторингу місцезнаходження розглядалися Козловським Е. М., який надав характеристики можливих моделей та способів отримання даних геолокації та описав складнощі і обмеження таких систем [5]. Кухтій А. та Кухтій С. у своїх роботах обґрунтували доцільність використання інформаційних технологій в навігації для пересічних користувачів та бізнесу [1].

Багатоканальними системами, аналого-цифровими (АЦ) перетвореннями, опрацюванням і аналізом даних з акустичних сенсорів в тім числі й у сейсморозвідці займається Крупельницький Л. В. [2]. Ефективним вирішенням задач цифрової обробки сигналів з використанням кодів золотої пропорції займається Черняк О. І. [6].

Великий інтерес до навігації та геолокації є у компаній, таких як: Google, SkyRiver, EasyWay [7-10]. Проблемами моніторингу місцезнаходження та навігації займається компанія Google, що надає можливості та сервіси щодо карт і геолокації [7-8]. Компанія SkyRiver - дозволяє спостерігати за місцезнаходженням транспортного засобу або стаціонарного об'єкту з будь-якого сучасного мобільного пристрою [9]. Також варто відзначити український проект для пошуку маршрутів громадського транспорту EasyWay, що допомагає підібрати зручний маршрут з урахуванням фінансових та часових затрат [10].

Питаннями, що покладено в основу статті займались провідні фахівці як в нашій державі, так і за кордоном. Фізичні та юридичні особи, згадані вище зробили значний вклад у розвиток тем геолокації, навігації, сейсморозвідки, тощо. Але в існуючих наукових дослідженнях та розробках в цілому не вирішуються питання опрацювання даних дистанційно-розподілених систем та пошуку об'єктів на місцевості, та вимагається впровадження сучасних мережевих технологій збору, збереження та опрацювання даних.

Структура і характеристики програмного забезпечення

Структурна організація ПЗ для опрацювання даних дистанційно-розподілених систем та пошуку об'єктів на місцевості зображена на рис. 1.

ПЗ для опрацювання даних дистанційно-розподілених систем та пошуку об'єктів на місцевості розроблено з використанням сучасного стеку технологій: Java, Spring Framework, Maven, MongoDB, Swagger, Docker, Google Maps Services. Обраний стек є добре задокументованим та вільним у використані. що дозволяє зробити ПЗ надійним, незалежним від платформи та відносно дешевим [11].

Розроблене ПЗ дозволяє:

- знаходити та відображати на карті найближчі до місцезнаходження користувача об'єкти;
- здійснювати пошук відносно заданої координати та радіусу пошуку;
- здійснювати пошук в межах певної місцевості (населеного пункту, району і т.д.);
- відображати межі заданої території та де здійснюється пошук в реальному часі;
- отримувати усі типи об'єктів, передбачені сервісом Google Places API, наприклад ресторани, аеропорти, кінотеатри, а також користувацькі об'єкти, такі як: сейсмометри, БПЛА (за умови інтеграції з системою);
- визначати координати для наступної області пошуку, для розміщення сейсмографу, або ж координату та напрямок для переміщення БПЛА.

Користувачі мають змогу використовувати функціональні можливості даного ПЗ через Web-інтерфейс, або ж інтегрувати їх у свої розробки через API. Оскільки, функціонал реалізовано у вигляді мікро-сервісу, то існує можливість об'єднати його з іншими мікро-сервісами, що також мають своє API. Це дозволяє розширити кількість функціональних можливостей зібраних в інтерактивному Swagger документі.

На рисунку 1 зображено декілька сервісів, а саме CGE та CGP. CGE – абревіатура від City Geo Explorer. Це власне і є ПЗ для опрацювання даних дистанційно-розподілених систем та пошуку об'єктів на місцевості. Інший сервіс зображений на рисунку CGP – абревіатура від Codes of Golden Proportion. Це сервіс для віддаленого виконання арифметичних і логічних операцій в кодах золотої пропорції, якому відповідають публікації [12, 13, 14]. Для обробки даних дистанційно-розподілених систем (БПЛА та сейсмометрів обладнаних GPS-трекерами) стане у нагоді простота виконання арифметичних і логічних операцій та надлишковість кодів золотої пропорції. Це створює можливості для ряду напрямків, зокрема, підвищенні точності, синхронізації, контролю й діагностики [12]. Тому вирішено використати наявні напрацювання для віддаленого виконання арифметичних і логічних операцій в кодах золотої пропорції і об'єднати їх з тематикою цієї статті. Тим більше, що для керування запитами, збору документації з сервісів та авторизації користувачів потрібен API-шлюз, що також розроблено в попередніх публікаціях.

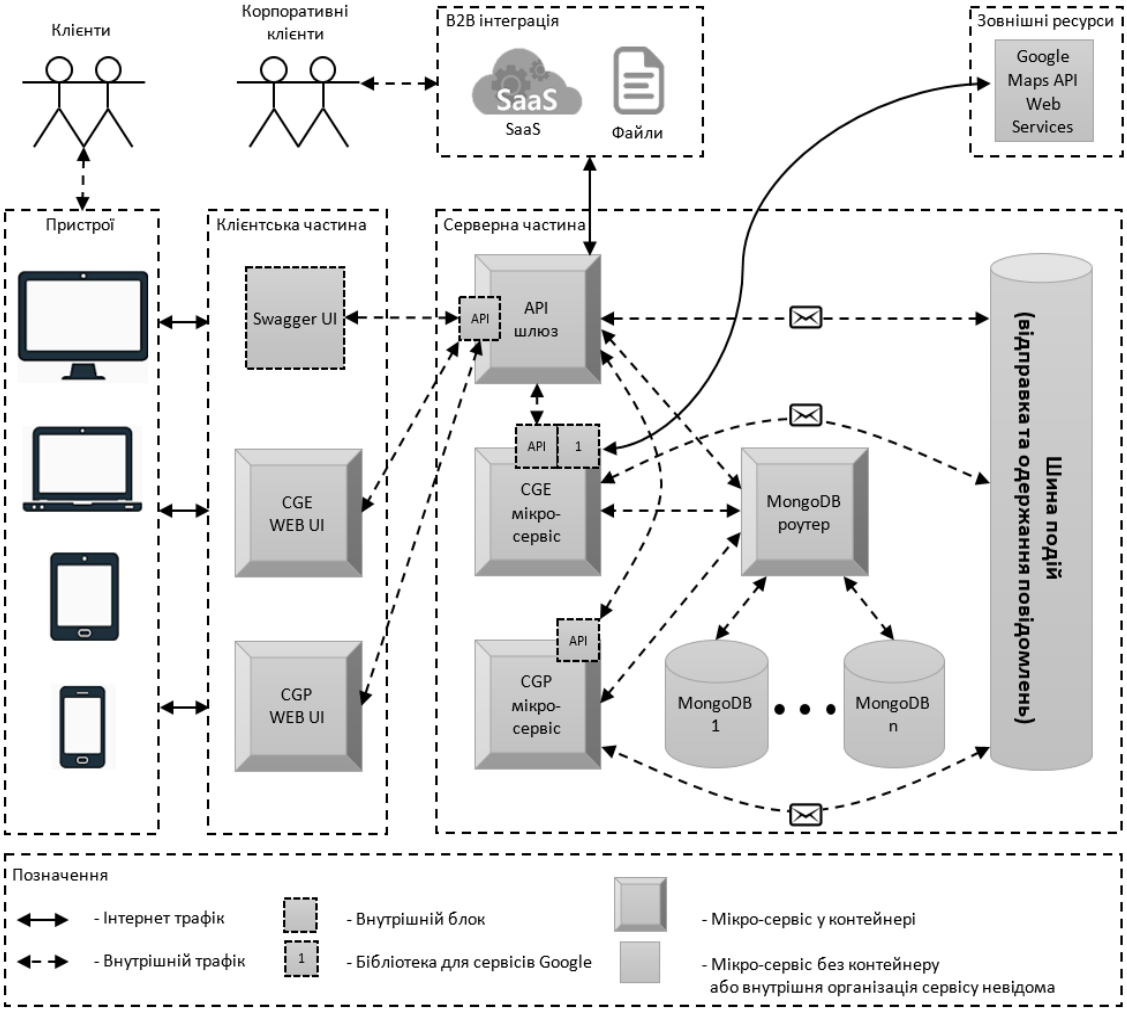


Рисунок 1 – Структурна організація розробленого ПЗ для опрацювання даних дистанційно-розподілених систем та пошуку об'єктів на місцевості

API-шлюз та CGP мікро-сервіс розроблено з використанням Scala, PlayFramwork, SBT, MongoDB, Swagger, Apache Kafka, що дещо відрізняється від основного стеку ПЗ для опрацювання даних дистанційно-розподілених систем та пошуку об'єктів на місцевості. Спільним є мікро-сервісна архітектура, база даних MongoDB та генератор інтерактивної документації Swagger. Відмінності в стеку не є проблемою, адже сервіси ізольовані і комунікують лише за допомогою http запитів, json файлів, обмінюються повідомлення через Kafka чи RabbitMQ. Принцип комунікації між сервісами за допомогою Kafka чи RabbitMQ зображено на рис. 2.

Комунікація через такі засоби як Kafka або RabbitMQ необхідна тому, що ці засоби дозволяють створювати чергу повідомлень, які поступово будуть оброблятись. У випадку, якщо якийсь сервер був тимчасово недоступний, обробка буде продовжена з того місця де зупинилися. Теми можуть мати багато розділів і в кожного свій власний індекс з якого потрібно продовжити обробку. Відправник може надсилати повідомлення одразу на декілька тем, або розділів. Так само одержувачів підписаних на одну тему може бути багато, щоб обробити необхідну кількість даних. Все це не можливо при використанні звичайних http запитів до сервісів.

Як видно з рис. 1, сервіси ізольовано у контейнерах. Для цього використовується контейнерна віртуалізація на базі Docker. Відмінності обраного способу віртуалізації від звичайної віртуалізації зображено на рис. 3.

Використання контейнерної віртуалізації дає такі переваги:

- додатковий захист. Кожен сервіс безпечно ізольовано в окремому контейнері. На зовні ми віддаємо, лише те, що потрібно;
- сервіси можуть використовувати різні версії залежностей і не конфліктувати між собою;
- Docker надає в користування безліч готових конфігурацій та образів для підняття контейнерів;

• завдяки технології контейнерної віртуалізації, що працює без додаткового навантаження гіпервізора і дозволяє спільно використовувати ресурси батьківської ОС, можна запускати багато контейнерів на одному сервері одночасно;

• легко обмінюватись готовим сконфігурованим контейнером та запускати його на різних серверах з різною апаратурою та ОС;

• контейнери можуть працювати на локальній машині, як реальній так і на віртуальній машині в дата центрі чи в хмарі;

можна піднімати декілька екземплярів сервісів і розподіляти навантаження між ними.

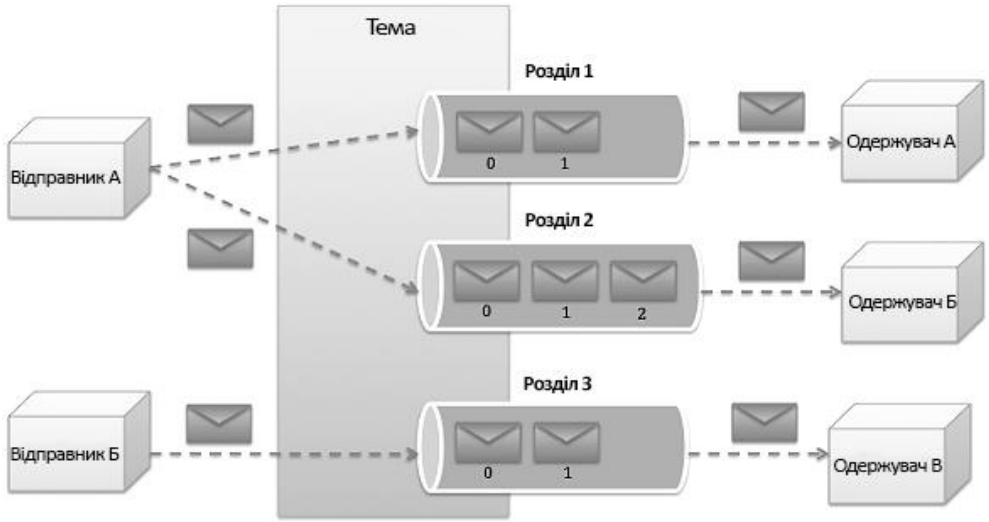
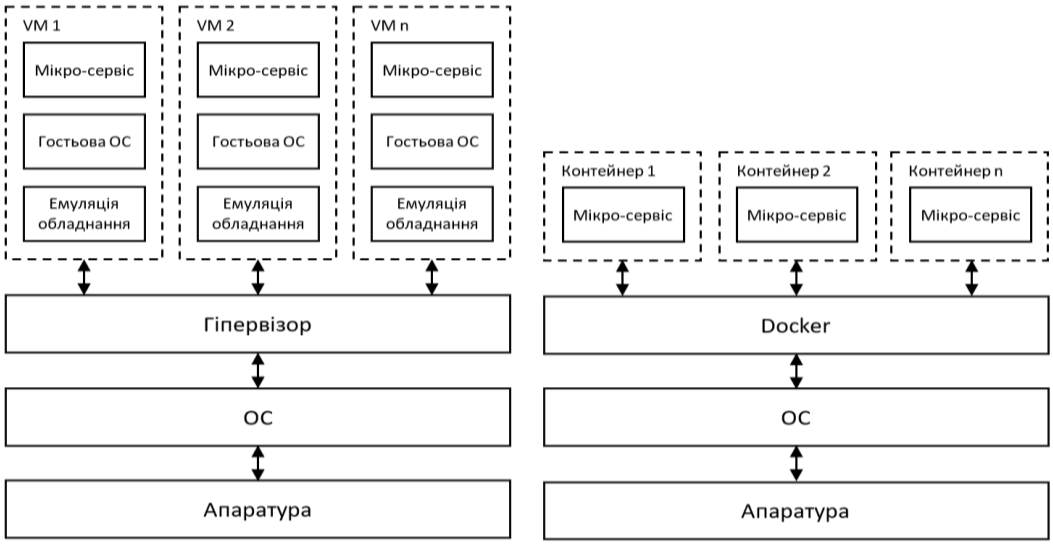


Рисунок 2 – Комунікація між сервісами



Звичайна (гіпервізорна) віртуалізація Контейнерна Docker віртуалізація

Рисунок 3 – Порівняння контейнерної та звичайної віртуалізації

•

На рис. 1 зображено декілька баз даних і роутер. Це пояснюється тим, що база даних спроектована для обробки великої кількості даних із забезпеченням високої продуктивності. Це можливо завдяки горизонтальному масштабуванню. В термінології баз даних це називають шардингом. Щоб не перевантажувати один сервер ми розподіляємо інформацію на декілька серверів, а роутер визначає до якої бази даних звернутись за потрібними даними.

Завдяки такому способу організації ПЗ маємо ряд суттєвих переваг та можливостей [14]:

• можливість інтегрувати сервіси в кластер мікро-сервісів з різним функціоналом призначеним для моделювання різних процесів, обробки та аналізу даних отриманих з АЦ систем;

• можливість працювати з будь-якого пристрою, що має доступ до мережі Інтернет;

- можливість розгортання розробленої системи в хмарній інфраструктурі;
- можливості аналітики з інформацією про те, що відбувається в будь-якому сегменті багатовимірного і багатоаспектного ланцюжка сервісів у рамках цифрового продукту або послуги;
- можливість розповсюджувати програмний продукт за принципом Software as a Service (SaaS), де замовники платять не за володіння програмами як такими, а за їх використання;
- можливість створити модель партнерської цифрової екосистеми, а також спрощення інтеграції продуктів між собою.

Висновки

ПЗ розроблено з використанням сучасного стеку технологій: Java, Spring Framework, Maven, MongoDB, Swagger, Docker, Google Maps Services та є надійним, незалежним від платформи та відносно дешевим. Користувачі мають змогу використовувати функціональні можливості через Web-інтерфейс, або ж інтегрувати функціонал у свої розробки через спеціальне API.

Створене ПЗ для опрацювання даних дистанційно-розподілених систем та пошуку об'єктів на місцевості дозволяє:

- знаходити та відображати на карті найближчі до місцезнаходження користувача об'єкти;
- здійснювати пошук відносно заданої координати та радіусу пошуку;
- здійснювати пошук в межах певної місцевості (населеного пункту, району і т.д.);
- відображати межі заданої території та де здійснюється пошук в реальному часі;
- отримувати усі типи об'єктів, передбачені сервісом Google Places API, наприклад ресторани, аеропорти, кінотеатри, а також користувацькі об'єкти, такі як: сейсмометри, БПЛА (за умови інтеграції з системою);
- визначати координати для наступної області пошуку, для розміщення сейсмографу, або ж координату та напрямок для переміщення БПЛА.

Завдяки запропонованій архітектурі ПЗ маємо ряд суттєвих переваг та можливостей:

- можливість інтегрувати сервіси в кластер мікро-сервісів з різним функціоналом призначеним для моделювання різних процесів, обробки та аналізу даних отриманих з АЦ систем;
- можливість працювати з будь-якого пристрою, що має доступ до мережі Інтернет;
- можливість розгортання розробленої системи в хмарній інфраструктурі;
- можливості аналітики з інформацією про те, що відбувається в будь-якому сегменті багатовимірного і багатоаспектного ланцюжка сервісів у рамках цифрового продукту або послуги;
- можливість розповсюджувати програмний продукт за принципом SaaS, де замовники платять не за володіння програмами як такими, а за їх використання;
- можливість створити модель партнерської цифрової екосистеми, а також спрощення інтеграції продуктів між собою.

Література

[1] А. Кухтій, та С. Кухтій, Формування туристичних маршрутів з використанням сучасних інформаційних технологій. Львів, Україна: ЛДУФК, 2014, 242 с.

[2] Л. В. Крупельницький, «Характеристики і структури багатоканальних АЦ-систем, що самокорегуються, для аналізу аудіо сигналів,» на V Міжнар. наук.-практ. конф. Методи та засоби кодування, захисту й ущільнення інформації, Вінниця, 2016, с. 129-133.

[3] Arkadiy Sedov, «Огляд сфер використання БПЛА в повсякденному житті», [Електронний ресурс] – Режим доступу: http://www.50northspatial.org/ua/uavs-everyday-life/. Дата звернення: 26.08.18.

[4] И. А. Липкин, Спутниковые навигационные системы. Москва: Россия: Вузовская книга, 2001, 86 с.

[5] Е. М. Козловский, Искусство позиционирования. Москва, Россия: Вокруг света, 2006, 280 с.

[6] О. І. Черняк, «Потокові методи і засоби повнофункціональної побітової арифметики зі зменшеними витратами обладнання,» дис. канд. техн. наук., фак-т інфор. техн. і комп. Інженер., Він. нац. техн. ун-т, Вінниця, 2013. - 20 с.

[7] Developer's Guide [Електронний ресурс]. – Режим доступу: https://developers.google.com/api-client-library/java/google-api-java-client/dev-guide. Дата звернення: 26.08.18.

[8] Java client library for Google Maps API Web Services [Електронний ресурс]. – Режим доступу: https://github.com/googlemaps/google-maps-services-java. Дата звернення: 26.08.18.

[9] SkyRiver [Електронний ресурс]. – Режим доступу: http://skyfleet.com.ua/.

[10] Система пошуку маршрутів громадського транспорту EasyWay [Електронний ресурс]. – Режим доступу: https://www.eway.in.ua/. Дата звернення: 26.08.18.

[11] О. Д. Азаров, Л. В. Крупельницький, О. І. Черняк, та В. В. Залізецький «Система дистанційної колективної самопідготовки,» Інформаційні технології та комп'ютерна інженерія. №2(36), с. 15-20, 2016.

[12] О. Д. Азаров, О. І. Черняк, В. В. Залізецький «Програмне забезпечення для віддаленого виділення цілої і дробової частин чисел у кодах золотої пропорції,» на VI Міжнар. наук.-практ. конф. Методи та засоби кодування, захисту й ущільнення інформації, Вінниця, 2017. – с. 163-166.

[13] Залізецький В. В. «Програмне забезпечення для віддаленого виконання арифметичних і логічних операцій в кодах золотої пропорції,» Матеріали конференції Молодь в науці: дослідження, проблеми, перспективи (МН-2018) [Електронний ресурс], Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2018/paper/view/3756. Дата звернення: 26.08.18.

[14] Залізецький В. В. «Програмне забезпечення для віддаленого додавання та віднімання кодів золотої пропорції в порозрядному режимі,» Матеріали XLVII науково-технічної конференції підрозділів [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2018/paper/view/5269. Дата звернення: 26.08.18.

Стаття надійшла: 27.08.18.

Відомості про авторів

Азаров Олексій Дмитрович, д.т.н., професор, декан факультету інформаційних технологій та комп'ютерної інженерії Вінницького національного технічного університету, заслужений працівник освіти України.

Черняк Олександр Іванович, к. т. н., доцент кафедри обчислювальної техніки Вінницького національного технічного університету; адреса: 21021.

Залізецький Василь Володимирович, аспірант кафедри обчислювальної техніки Вінницького національного технічного університету.

O. D. Azarov[1], O. I. Chernyak[1], V. V. Zalizetskiy[1]

SOFTWARE FOR PROCESSING DATA OF DISTANCE-DISTRIBUTED SYSTEMS AND SEARCHING FOR OBJECTS ON A LOCATION

[1]Vinnytsia National Technical University

УДК 004.7:519

С. М. Захарченко[1], К. І. Шевчук[1]

МЕТОД ВДОСКОНАЛЕННЯ ОДНОШЛЯХОВИХ ПРОТОКОЛІВ ДИНАМІЧНОЇ МАРШРУТИЗАЦІЇ

[1]Вінницький національний технічний університет

Анотація. Розглядаються сучасні протоколи динамічної маршрутизації, принцип їх функціонування та способи призначення метрики зв'язків, визначено, що більшість з них є одношляховими, обирають один маршрут з мінімальною метрикою або здійснюють балансування між маршрутами з однаковою метрикою. Як результат функціонування мережі у такий спосіб спричиняє максимальне використання знайденого найкращого або альтернативного шляху, їх перевантаження, в той час як інші вузли (ресурси) мережі не задіяні при передачі трафіку. Оскільки подібне рішення є характерним для всіх одношляхових протоколів, впровадження змін безпосередньо в самому протоколі є нераціональним. Вирішення цієї проблеми можливо здійснити через структурні зміни в мережі, а саме модифікацію процесу маршрутизації, скомбінувавши найкращі особливості розподіленого та централізованого способів маршрутизації. Протоколи маршрутизації продовжують працюють за розподіленим принципом – кожний маршрутизатор самостійно будує таблиці маршрутизації на основі інформації, що отримана від інших маршрутизаторів. Після впровадження в мережу централізованого контролера маршрутизації, від всіх маршрутизаторів періодично надходитиме інформація про їх функціональний стан. Контролер при виникненні потреби через протокол SNMP вносить зміни в параметри маршрутизатора з подальшою зміною таблиць маршрутизації, а модуль керування проводить аналіз та розрахунки для ініціювання змін.

Ключові слова: трафік, протокол, динамічна маршрутизація, метрика, контролер, модуль керування, балансування, альтернативні маршрути, оптимізація потоку.

Аннотация. Рассматриваются современные протоколы динамической маршрутизации, принцип их функционирования и способы назначения метрики связей, определено, что большинство из них являются однопутевыми, выбирают один маршрут с минимальной метрикой или осуществляют балансировку между маршрутами с одинаковой метрикой. Функционирование сети таким образом вызывает максимальное использование найденного лучшего или альтернативного пути и его перегрузки, в то время как другие узлы (ресурсы) сети не задействованы при передаче трафика. Поскольку подобное решение характерно для всех однопутевых протоколов, внедрение изменений непосредственно в самом протоколе является нерациональным. Решение этой проблемы возможно осуществить через структурные изменения в сети, а именно модификацию процесса маршрутизации, скомбинировав лучшие особенности распределенного и централизованного способов маршрутизации. Протоколы маршрутизации продолжают работу по распределенному принципу - каждый маршрутизатор самостоятельно строит таблицы маршрутизации на основе информации, полученной от других маршрутизаторов. После внедрения в сеть централизованного контроллера маршрутизации, от всех маршрутизаторов периодически поступает информация об их функциональном состоянии. Контроллер при возникновении потребности через протокол SNMP вносит изменения в параметры маршрутизатора с последующим изменением таблиц маршрутизации, а модуль управления проводит анализ и расчеты для инициирования изменений.

Ключевые слова: трафик, протокол, динамическая маршрутизация, метрика, контроллер, модуль управления, балансирование, альтернативные маршруты, оптимизация потока.

Abstract. The modern protocols of dynamic routing, the principle of their functioning and methods of assigning a link metric are considered, it is determined that most of them are one-way, choose one route with a minimum metric or carry out balancing between routes with the same metric. As a result, the network operation in this way causes the maximum use of the best or alternative path found, their overload, while other nodes (resources) of the network are not involved in the transmission of traffic. Since such a decision is necessary for all one-way protocols, the introduction of changes directly to the protocol itself is not rational. The solution to this problem may be due to structural changes in the network, namely the modification of the routing process, combining the best features of distributed and centralized routing methods. Routing protocols continue to work according to the distributed principle - each router independently builds routing tables based on information received from other routers. After the implementation of the centralized router controller, all routers will periodically receive information about their functional state. When required by the controller, the SNMP changes the router's parameters with the subsequent change of the routing tables, and the control module conducts the analysis and calculations to initiate the change.

Key words: traffic, protocol, dynamic routing, metric, controller, control module, balancing, alternative routes, stream optimization.
DOI: https://doi.org/10.31649/1999-9941-2018-42-2-16-25.

Вступ

Протоколи маршрутизації визначають способи взаємодії маршрутизаторів та функціонують з метою визначення найкращих маршрутів при передачі трафіку між вузлами мережі. Із зростанням кількості користувачів, рівня важливості переданої інформації, специфіки умов експлуатації мереж, впровадження нових послуг, відбулися зміни у властивостях самих мереж передачі інформації. Актуальним став пошук нових підходів до розвитку мереж, і, що не менш важливо, з'явилась необхідність надати сучасним мережевим технологіям нових властивостей.

Для підвищення ефективності роботи мережі та підвищення якості надаваних послуг, виникла необхідність використовувати облік стану мережі [1]. В перспективі вирішення такої задачі можливо здійснити за умови розвитку та застосування методів розподілення інформації [2, 3]. Перевага розподілених алгоритмів в тому, що їх налаштування відбувається в одному часовому масштабі, і вони мають здатність швидко реагувати на зміни трафіку [4].

В роботі [5] проведено узагальнення та класифікація методів багатошляхової маршрутизації з урахуванням специфіки передачі трафіку по декільком шляхам, а також традиційних підходів до класифікації всіх алгоритмів маршрутизації. При аналізі методів маршрутизації є очевидним, що одношляхові

протоколи маршрутизації, які для пошуку найкоротшого шляху використовують класичні алгоритми Дейкстри, Беллмана-Форда, Шуурбале, не можливо використовувати як спосіб балансування навантаження (трафіку) в мережі, оскільки їх специфіка полягає у передачі трафіку по одному – найкращому маршруту. Крім того в більшості випадків шлях обираються без урахування поточного завантаження інших ресурсів мережі. Якщо найкоротший шлях уже перевантажений, то пакети всеодно надсилатимуться саме цим шляхом, що погіршуватиме ситуацію в мережі.

Алгоритми використання резервних каналів дозволяють використовувати окрім основного додатковий шлях для передачі трафіку. Реалізація ідеї виділення окремого резервного каналу розглянута у [6, 7] і відображає часткове вирішення проблеми одношляхових протоколів маршрутизації. Слід зазначити, що за наявності в мережі декількох альтернативних чи резервних маршрутів, трафік пропорційно розподіляється між ними, і навантаження на маршрутизатори та канали зв'язку розподіляються більш збалансовано. Проте такий підхід не завжди є доцільним, особливо тоді, коли такі маршрути не є повністю рівноцінними, тобто мають різну метрику.

Одним із найбільш перспективних способів керування трафіком в мережах є модель програмно-керованої мережі (SDN), яка передбачає розподіл функцій передачі трафіку і функцій керування, включаючи контроль як самого трафіку так і пристроїв, які здійснюють його передачу. Згідно концепції SDN, вся логіка керування розташована у контролерах, які здатні відслідковувати роботу всієї мережі за допомогою спеціальних протоколів (наприклад, OpenFlow), які оперують поняттям "потоки" (flow) і можуть здійснювати різні дії з ними (дозволяти, забороняти, перенаправляти, змінити поля в пакетах і т. д.). Перевагами програмно-керованої мережі є централізоване управління, спрощення обслуговування та модернізації мереж, скорочення часу на оновлення програмних кодів комутаторів/маршрутизаторів і впровадження нових сервісів. Однак є певні недоліки, пов'язані з архітектурою Open SDN та способами практичного впровадження таких мереж. Вважається, що ця технологія несе надто радикальні зміни в функціонування мереж [8]. Це виявляється в обов'язковому встановленні вартісного обладнання з підтримкою інтерфейсу OpenFlow. Найбільш серйозним недоліком функціонування програмно-керованої мережі за технологією SDN є вразливість такої мережі через існування однієї точки відмови, мається на увазі ті випадки, коли виходить з ладу контролер, мережа повністю втрачає працездатність. SDN функціонує за принципом централізованої маршрутизації. З метою усунення цього недоліку, необхідно впроваджувати в мережу декілька контролерів, котрі будуть взаємодіяти один з одним за допомогою надійних каналів зв'язку, що може значно ускладнити та збільшити у вартості архітектуру Open SDN.

Актуальність

Моделі багатошляхової маршрутизації з балансуванням навантаження (8) - (13), розглядаються як досить успішна реалізація принципів інжинірингу трафіку. Розподіл навантаження по великій кількості попередньо сформованих шляхів передачі даних між вузлами відправника та отримувача і є пріоритетнішим всіх завдань.

В мережі може виникати ситуація, коли загальний трафік, що передається по вибраному шляху, тимчасово перебільшує пропускну спроможність каналу, що обумовлено пульсуючим характером трафіку сучасних IP-мереж. Застосування технологій профілювання (policing) та формування (shaping) частково дозволяють вирішити проблему, однак передбачають відкидання частини пакетів, або затримку їх в буферній пам'яті. Більш ефективним шляхом уникнення перевантаження є поділ трафіку на частини і перенаправлення частини пакетів по інших шляхах.

Разом з тим, характер подій в мережі в певний часовий проміжок залежить від попередніх віддалених подій. Це означає, що при великих масштабах мережі трафік наділений властивостями подібності, тобто виглядає майже однаково при достатньо великих масштабах часового проміжку [14]. Це дозволяє пеним чином прогнозувати події в мережі і приймати рішення заздалегіть.

Інший недолік багатошляхової маршрутизації полягає в тому, що максимальне покращення балансування по каналам зв'язку можливе не для всіх топологій мережі. Особливо критичним зниженням якості балансування відбувалось у мережах з неоднорідною топологією, представлених у вигляді розподіленого графа. Для мінімізації цього недоліку пропонується замінити централізоване керування методом маршрутизації «по підмережам», оскільки повна ефективність балансування трафіку напряму залежить від якості балансування у «вузьких місцях» мережі, особливо це стосується неоднорідних мереж у вигляді розподіленого графа[14].

Слід зауважити, що більшість досліджень, спрямованих на вирішення проблем при передачі інформації в мережах мають здебільше теоретичний характер, в той час як в сучасних мережах застосовується обмежений перелік протоколів маршрутизації, більшість з яких є одношляховими. Саме тому тематика статті, присвячена методу вдосконалення одношляхових протоколів динамічної маршрутизації є доцільною та актуальною.

Мета

Метою дослідження є покращення роботи сучасних мереж за рахунок зменшення затримки передачі трафіку шляхом модифікації процесу маршрутизації без внесення змін в існуючі протоколи динамічної маршрутизації.

Задачі

1. Проаналізувати сучасні протоколи динамічної маршрутизації та визначити потенційні шляхи покращення їх роботи.
2. Розробити комбінований розподілено-централізований протоколонезалежний варіант динамічної маршрутизації.
3. Розробити алгоритм роботи контролера маршрутизації та модуля керування.
4. Оцінити результат роботи мережі після впровадження змін.

Аналітичний огляд методів динамічної маршрутизації

Для вирішення задачі підвищення якості передачі трафіку в мережі необхідно проаналізувати методи динамічної маршрутизації, їх функціональні особливості, основні переваги та недоліки з метою визначення ймовірних негативних явищ в мережі та методи впливу на них.

За способом маршрутизації мережі бувають з централізованою, децентралізованою та гібридною маршрутизацією.

Централізована маршрутизація реалізується за принципом вибору напрямку руху для кожного пакету центром управління мережею, а мережеві вузли лише сприймають та реалізують результати вирішення задачі маршрутизації. Перевагами такого способу є можливість обрати вузли прості за структурою, оскільки вони беруть мінімальну участь в процесі маршрутизації. Однак при збільшенні кількості вузлів зростає складність організації централізованого керування мережі передачі даних. Суттєвим недоліком централізованого керування є пряма залежність якості маршрутизації від надійності її центру керування, яка із збільшенням складності останнього має тенденцію до зниження. Крім того, центр керування мережі повинен мати оперативну інформацію про стан мережі, тому що вихід з ладу вузла або його перевантаження може спричинити втрату працездатності всієї мережі.

Розподілена або децентралізована маршрутизація виконується через розподіл функцій керування мережею між її вузлами. На основі збереженої керуючої інформації кожен вузол самостійно визначає напрямок передачі пакетів. Це підвищує структурну складність вузлів, однак мережа відзначається вищим рівнем працездатності, оскільки вихід з ладу будь-якого вузла не впливає на роботу мережі загалом.

Гібридна маршрутизація характеризується застосуванням принципів централізованої та розподіленої маршрутизації (наприклад, гібридна адаптивна маршрутизація). Адаптивна маршрутизація передбачає пристосування алгоритму маршрутизації до реального стану мережі. Недоліком методів адаптивної маршрутизації є складність прогнозування стану мережі.

За кількістю визначених маршрутів до одного адресата протоколи маршрутизації поділяють на одношляхові (single-path) та багатошляхові (multi-path). Одношляхові протоколи заносять до таблиці маршрутизації дані про єдиний оптимальний маршрут. Очевидним недоліком є нерівномірне навантаження мережі через максимальне завантаження оптимального маршруту. Багатошляхові протоколи вирізняються визначенням декількох оптимальних шляхів. Це дозволяє розпаралелити передачу трафіку і, як наслідок, збільшити надійність передавання даних та ефективність використання каналів зв'язку. Не зважаючи на очевидні переваги багатошляхових протоколів на сьогоднішній день в сучасних мережах застосовуються одношляхові, найбільш відомими з яких є OSPF та EIGRP.

OSPF (Open Shortest Path First) - це широко використовуваний протокол внутрішнього шлюзу (IGP), заснований на технології відстеження стану каналу (link-state technology) та пошуку найкоротшого шляху. Цей протокол здійснює маршрутизацію пакетів, збираючи інформацію про стан зв'язків з сусідніх маршрутизаторів і на підставі отриманої інформації будує карту мережі. Маршрутизатори OSPF надсилають багато типів службових повідомлень, включаючи повідомлення hello, запити на стан зв'язку, оновлення та опис бази даних. Пошук найкоротшого шляху здійснюється за алгоритмом Дейкстри. Для вибору найкращого маршруту OSPF використовує метрику (cost), яка за замовчуванням розраховується на основі смуги пропускання каналу.

Переваги передачі трафіку при застосуванні OSPF в тому, що зміни топології мережі відпрацьовуються дуже швидко. Головним недоліком протоколу OSPF є те, що завдяки використанню алгоритму Дейкстри визначається один найкращий маршрут, за яким і спрямовується увесь трафік. Це може призводити до перевантажень в IP-мережі і вимагає реалізації додаткових методів. Наприклад, в [14] пропонується використовувати в критерії розподілу залишкову пропускну здатність каналу.

EIGRP (Enhanced Interior Gateway Routing Protocol) - дистанційно-векторний протокол динамічної маршрутизації, що був оптимізований для зменшення нестабільності протоколу після змін топології мережі, уникнення проблеми зациклення маршруту та більш ефективного і економного використання по-

тужностей маршрутизатора та пропускної спроможності ліній зв’язку. Композитна метрика, яка використовується для пошуку оптимального шляху, розраховується на основі пропускної здатності, навантаження, затримки та надійності. Це дозволяє підвищити якість вибору оптимального маршруту.

Основними перевагами EIGRP є: низьке споживання мережевих ресурсів за відсутності змін в топології (передаються тільки пакети "hello"); при виникненні змін по мережі передається тільки інформація про модифікації, які відбулись, що дозволяє зменшити навантаження на мережу та забезпечує малий час конвергенції (в окремих випадках збіжність забезпечується майже миттєво).

Поруч з перевагами сучасних протоколів динамічної маршрутизації, слід відзначити, що всі вони здійснюють пошук одного найкращого маршруту з мінімальною метрикою, тобто є одношляховими, або здійснюють балансування маршрутів у мережі з однаковою метрикою, що спричиняє максимальне використання знайденого найкращого або альтернативного шляху та його перевантаження, в той час як інші вузли (ресурси) мережі не будуть задіяні в процесі передачі трафіку. Такий підхід не дає можливості досягти стану повноцінної рівноваги, балансованого розподілу навантаження між всіма можливими альтернативними шляхами.

В протоколі EIGRP передбачено механізми реалізації багатошляхової маршрутизації зокрема за допомогою техніки балансування між шляхами з різною вартістю (unequal cost load balancing), однак він рідко застосовується, оскільки ускладнює процес налаштування. Крім того при розрахунку метрики в EIGRP за замовчуванням не використовуються такі динамічні параметри каналу як надійність і завантаженість, оскільки їх застосування призводить до постійних змін метрик і як наслідок перебудов маршрутів.

Виправити ситуацію через впровадження змін до конкретного протоколу є недоцільним, оскільки ця проблематика спостерігається в усіх протоколах динамічної маршрутизації, тому більш ефективним рішенням буде саме модифікація процесу маршрутизації без внесення змін в конкретний протокол маршрутизації. Такий варіант впливу дозволить універсально для всіх протоколів динамічної маршрутизації зменшити затримку передачі трафіку та збалансувати навантаження на мережу.

Принцип розподілено-централізованої маршрутизації з використанням контролера маршрутизації

Для успішного керування процесом маршрутизації необхідно знати стан кожного елемента мережі з можливістю змінювати параметри його функціонування. Зазвичай мережа складається з пристроїв різних виробників і здійснювати управління нею було б нелегким завданням, якби кожен з мережевих пристроїв розумів тільки свою систему команд. Саме тому було розроблено універсальний протокол управління мережевими ресурсами – SNMP (Simple Network Management Protocol).

SNMP реалізує такі функції:

- відправлення та прийом пакетів SNMP через протокол IP;
- збір інформації про статус і поточну конфігурацію мережевих пристроїв;
- зміна конфігурації мережевих пристроїв.

Крім управління пристроями SNMP використовують для моніторингу, оскільки він може отримувати різну інформацію від будь-яких мережевих пристроїв (маршрутизатор, комутатор або комп'ютер), в яких є підтримка даного протоколу. Вміст одержуваної інформації може бути різним, наприклад: час безперервної роботи, завантаженість центрального процесора, кількість пакетів в буфері маршрутизатора тощо.

Для вдосконалення роботи одношляхових протоколів маршрутизації пропонується здійснити структурні зміни в мережі, а саме модифікацію процесу маршрутизації шляхом впровадження в мережу централізованого контролера маршрутизації (рис. 1). Від всіх маршрутизаторів періодично або ситуативно надходитиме інформація про стан критичних параметрів. Контролер при виникненні потреби через протокол SNMP надсилає команди, що дозволяють внести зміни в параметри маршрутизатора, що, в свою чергу, впливає на маршрут проходження трафіку. Модуль керування виконує функцію допоміжного пристрою і в дуплексному режимі обмінюється інформацією з контролером. Функціонування модуля керування визначається різновидом протоколу маршрутизації а також вибраним способом впливу на трафік. Він проводить аналіз отриманих параметрів, розраховує порогові значення, необхідні зміни до метрик маршрутів, розмір завантаження буферу черги для перерозподілу тощо.

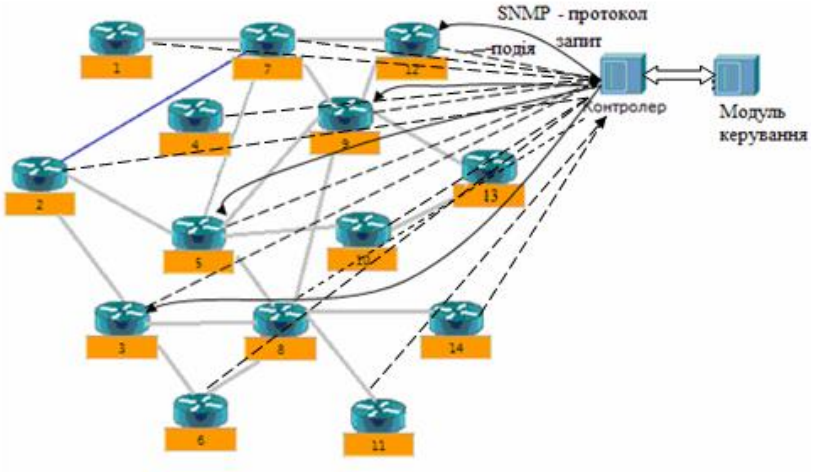


Рисунок 1 – Функціональна схема мережі після модифікації.

У випадку виходу з ладу контролера, маршрутизатори продовжують працювати автономно за допомогою протоколів динамічної маршрутизації, тобто недолік централізованого способу маршрутизації (втрата працездатності мережі через вихід з ладу центру управління) в модифікованому методі буде усунуто, як і недолік змішаного способу (складність прогнозування стану мережі), оскільки при працюючому контролері зміни виявляються миттєво через постійний моніторинг за допомогою протоколу SNMP.

Математична модель оцінювання якості процесу маршрутизації

Для оцінювання ефективності застосування запропонованого підходу скористаємось математичним апаратом систем масового обслуговування. Модель комп'ютерної мережі в цьому випадку складається з N-вузлів та L – каналів зв'язку. Трафік, що надходить в мережу створює пуассонівський потік з середнім значенням γ_{ij} (пакетів/сек) для пакетів, які надходять з вузла *i* до вузла *j* [16]. В такому випадку весь трафік буде визначатися як:

$$\gamma = \sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij} \tag{1}$$

Кожен канал зв'язку складається з дуплексного каналу з пропускною спроможністю d_{tf} (байт/с), де *t, f* – канали зв'язку між вузлами *t* та *f*. Якщо зв'язок між вузлами *t* та *f* відсутній, то $d_{tf} = 0$.

Визначимо $x_{tf}^{(i,j)}$ - як частину потоку γ_{ij}, що передається по каналу *t, f*: $0 \le x_{tf}^{(i,j)} \le 1$

Тоді:

$$\lambda_{tf} = \sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij} \cdot x_{tf}^{(i,j)} \tag{2}$$

де λ_{tf} - величина потоку в каналі *t, f* (пакетів/сек), визначена потоком γ_{ij}.

Коефіцієнт завантаження каналу ρ визначається як відношення величини потоку в каналі λ_{tf} до μd_{tf}, де $\mu d_{tf} = d_{tf} / N_{\text{пак}}$, $N_{\text{пак}}$ - середній розмір пакету:

$$\rho = \frac{\lambda_{tf}}{\mu d_{tf}} \tag{3}$$

Визначимо Z_{ij} як середній час, що витрачається на передачу повідомлення, яке створено на вузлі *i* для передачу вузлу *j* (затримка повідомлення). Важливою характеристикою якості функціонування мережі передачі даних є середня затримка пакетів в мережі – T, яка визначається як зважена сума затримок пакетів - Z_{ij}:

$$T = \frac{1}{\gamma}\sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij}\cdot Z_{ij} \tag{4}$$

Застосувавши до черг в мережі формулу Літтла, отримуємо: $T = \frac{1}{\gamma}\sum_{t=1}^{N}\sum_{f=1}^{N}\lambda_{tf}\cdot t_{tf}$, де t_{tf} - середній час перебування повідомлення в каналі *t, f*. Оскільки отримання аналітичного значення t_{tf} як і відповідно T не вбачається можливим, в такому випадку середній час перебування повідомлення в каналі *t, f,* що складається з часу передачі пакетів - $\frac{1}{\mu d_{tf}}$ та часу очікування в черзі - W_{tf}, визначається за формулою: $t_{tf} = \frac{1}{\mu d_{tf}} + W_{tf}$, де $W_{tf} = \frac{1}{\mu d_{tf}}\frac{\lambda_{tf}}{\mu d_{tf} - \lambda_{tf}}$ тоді : $t_{tf} = \frac{1}{\mu d_{tf} - \lambda_{tf}}$.

Визначимо $f_{tf} = \lambda_{tf} / \mu$ - величина потоку в каналі *t, f,* виражена в байт/с. Тоді: $t_{tf} = \frac{1}{\mu}\frac{1}{d_{tf} - f_{tf}}$.

Враховуючи всі підстановки отримуємо вираз для середньої затримки пакетів в мережі:

$$T = \frac{1}{\gamma}\sum_{t=1}^{N}\sum_{f=1}^{N}\frac{f_{tf}}{d_{tf} - f_{tf}} \tag{5}$$

Слід звернути увагу, що абсолютне значення, отримане за допомогою виразу (5) в загальному випадку не є вірним, оскільки не враховує низку особливостей передавання даних в сучасних мережах. З іншого боку цей вираз дає можливість досить просто порівняти між собою різні варіанти розподілення трафіку.

Узагальнений алгоритм роботи контролера маршрутизації та модуля керування.

На початку роботи контролер, на основі принципу роботи протоколу динамічної маршрутизації, наприклад OSPF, має зібрати всю інформацію про пристрої, якими буде здійснювати керування та наявність зв`язків між ними, побудувавши граф мережі. Перелік інформації, що збирається може відрізнятись для різних протоколів динамічної маршрутизації. Крім характерних параметрів, що розсилає той чи інший протокол динамічної маршрутизації через SNMP можуть бути запитані інші специфічні параметри, наприклад обсяги буферної пам'яті, значення MTU тощо.

На основі отриманої інформації формується граф мережі і визначаються параметри пристроїв, які в подальшому будуть аналізуватись та через які буде здійснюватися вплив на той чи інший мережевий пристрій. Зокрема має бути розрахована пропускна спроможність каналів d (байт/с), визначені критичні межі заповнення буферної пам'яті Q_{min}, Q_{max} та завантаження каналів ρ_{min}, ρ_{max}. Якщо вплив на вибір маршруту буде здійснюватися через вартості зв'язків, тоді необхідно визначити поточні метрики ліній - $Cost_R$ та максимальні метрики $Cost_{max}$, які в подальшому будуть використовуватись для розвантаження проблемного шляху. Також слід визначити величини відновлюваних метрик $Cost_{re}$ – на етапі перепризначення максимального значення поточним метрикам, значення останніх повинні бути збережені та в подальшому відновлені, оскільки метрики змінюються тимчасово.

Для оцінювання рівня завантаженості окремих ділянок може бути використано коефіцієнт завантаженості каналу ρ, що визначається виразом (1.3) або рівень заповнення вхідних черг маршрутизатора Q. Межі порогових значень ρ та Q залежать від вимог до якості обслуговування, типу трафіку тощо та потребують додаткових досліджень. Використання двох порогових значень дозволить уникнути частих перебудов маршрутів: так перенаправлення трафіку на альтернативний маршрут буде виконуватись при досягненні ρ_{max} на проблемній лінії, в той час як повернення до початкового варіанту при досягненні ρ_{min}. Для фіксації проблемних ситуацій на кожному маршрутизаторі налаштовуються порогові значення завантаженості каналу або заповнення буферу черг, досягнення яких має ініціювати відповідне SNMP повідомлення (SNMP-trap). Отримавши SNMP-trap та знаючи поточний стан завантаженості інших каналів, контролер переспрямовує трафік на інший маршрут, якщо такий знайдено. Визначення можливості повернення до початкового стану може бути реалізовано шляхом періодичного опитування (SNMP-

GetRequest). Узагальнену граф-схему алгоритму роботи контролера та блоку керування наведено на рис.2.

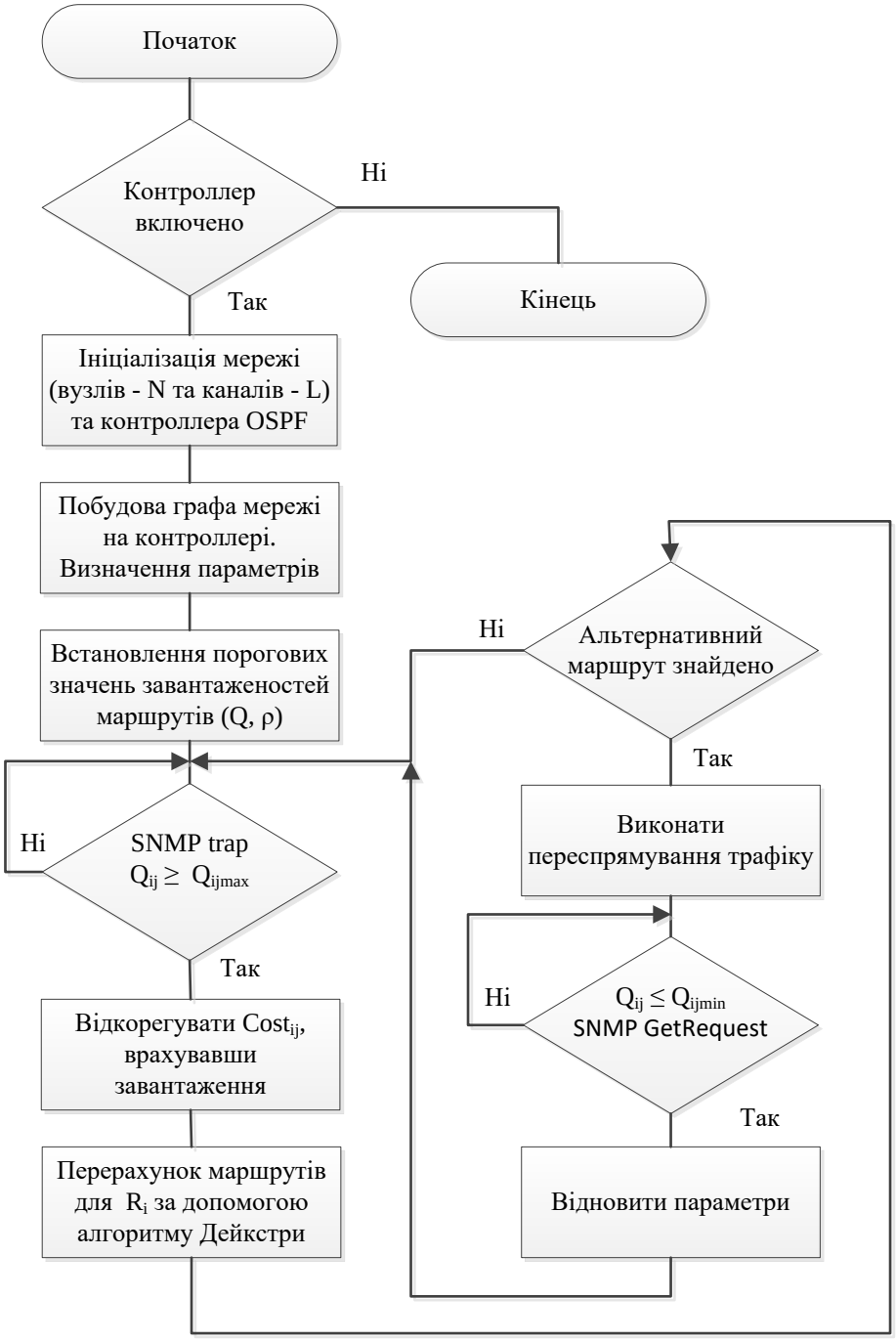


Рисунок 2 – Узагальнений алгоритм роботи контролера маршрутизації та модуля керування.

Оцінювання результатів застосування запропонованого методу

Розглянемо, як приклад, мережу наведену на рис. 3, яка складається з п’яти маршрутизаторів Ra, Rb, Rc, Rd та Re. Через мережу проходять два потоки трафіку: Ra-Rb-Rc – потік з постійною інтенсивністю 15 п/с та потік Ra-Rb-Rd-Re зі змінною інтенсивністю, яка зростає від 0 п/с до 210 п/с. Розмір пакету становить 1,5 КБ. Пропускні спроможності ліній зв’язку становлять: $d_{bc} = 2$ (МБ/с) та $d_{bd} = 2{,}5$ (МБ/с). Пропускні спроможності решти ліній будемо вважати значно більшими d_{bc}, що дозволить не враховувати їх в подальших розрахунках.

Припустимо, що на шляху Rb-Rd виникло перевантаження, Rb сповіщає контролеру маршрутизації про досягнення порогового рівня заповнення буферу черг Q_{max}. Контролер перенаправляє частину трафіку, що перебільшує порогове значення на інший маршрут Rb-Rc. Решта потоку буде проходити через цей маршрут до тих пір, доки значення завантаження буферу черг не зменшиться до нижнього порогового рівня Q_{min}. Отримавши відповідну інформацію контролер повертає решту потоку на маршрут Rb-Rd.

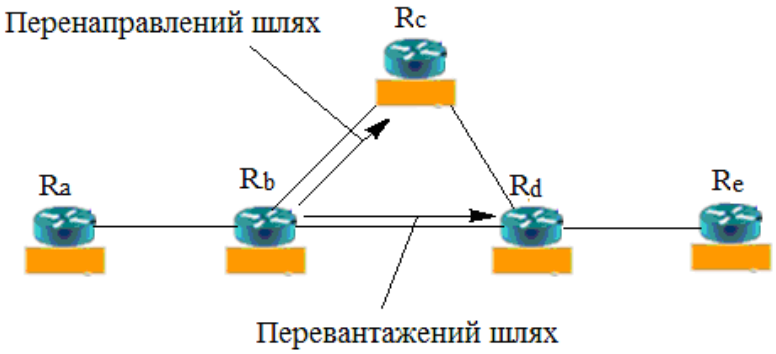


Рисунок 3 – Мережа з перевантаженим шляхом Rb-Rd

Розглянемо залежність середньої затримки передачі пакетів від інтенсивності потоку трафіку Ra-Rb-Rd-Re отриману на основі (1.5) і наведену на рис.4. Відповідно до протоколу маршрутизації OSPF для передавання потоку між маршрутизаторами Ra та Re буде вибрано маршрут Ra-Rb-Rd-Re, оскільки смуга пропускання лінії між Rb і Rd (2,5 МБ/с) більша за смугу між Rb і Rc (1,5 МБ/с). Крім того по маршруту Rb-Rc передається постійний трафік, який завантажує його приблизно на 15%.

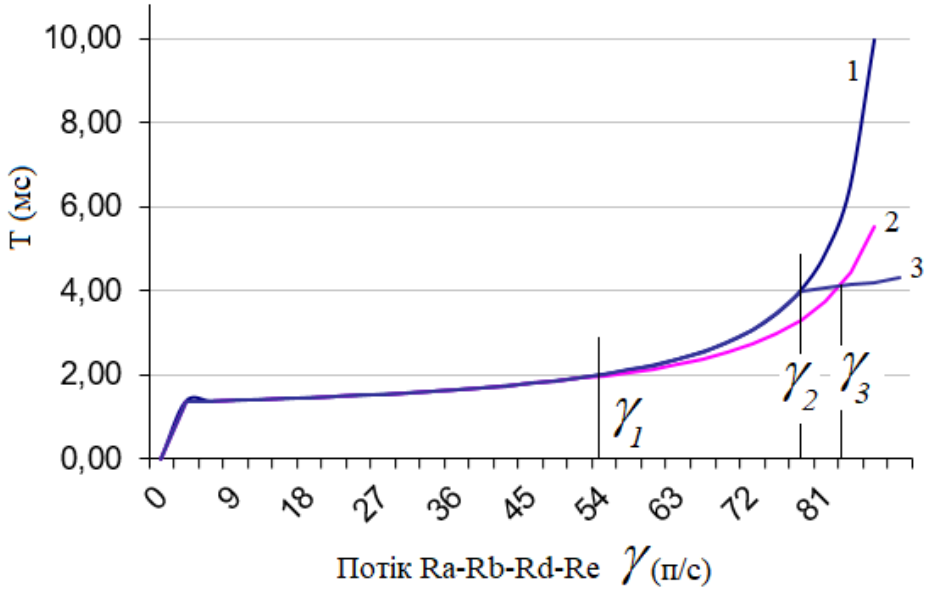


Рисунок 4 – Залежність середньої затримки передачі пакетів від значення потоку

При застосуванні класичного одношляхового підходу при зростанні трафіку Ra-Rb-Rd-Re середня затримка буди збільшуватись за експоненційним законом (лінія 1). Застосування запропонованого методу передбачає, що при досягненні певного ступеня завантаженості каналу Rb-Rd (на рис. 4 цей поріг досягається при значенні потоку γ_2), решта трафіку переспрямовується по маршруту Rb-Rc-Rd (лінія 3).

Вибір меншого порогу, наприклад γ_1, при якому починається переспрямування трафіку (лінія 2), дозволить зменшити середню затримку порівняно з попереднім випадком, однак при подальшому зростанні потоку призведе до перевантаження лінії Rb-Rc і починаючи зі значення γ_3 середня затримка буде більшою. Слід відзначити, що вибір оптимальних порогових значень для переспрямування трафіку зале-

жить від загальної структури схеми, пропускних спроможностей ліній зв'язку та потоків трафіку що передаються в мережі і потребує додаткових досліджень.

Висновки

1. Проведено аналітичний огляд сучасних протоколів динамічної маршрутизації, який показав що більшість з них є одношляховими, тобто вибирають один маршрут з мінімальною метрикою, або здійснюють балансування між маршрутами з однаковою метрикою, що спричиняє максимальне використання знайденого маршруту, його перевантаження, в той час як інші ресурси мережі будуть в стані очікування - не задіяні в процесі передачі трафіку.
2. Запропоновано комбінований розподілено-централізований метод маршрутизації, який за рахунок внесення змін в процес маршрутизації дозволяє уникати ситуацій перевантаження каналів, обумовлених логікою роботи одношляхових протоколів і не потребує внесення змін в сучасні протоколи динамічної маршрутизації, що спрощує процес його впровадження в сучасних мережах.
3. Відповідно до запропонованого методу розроблено алгоритм балансування навантаження для випадків перевантаження окремих каналів, реалізація якого дозволить зменшити навантаження на перевантажені канали і мінімізувати середню затримку доставки даних в мережі.
4. Визначено основні структурні елементи мережі для реалізації запропонованого методу, до яких входять маршрутизатори з підтримкою сучасних протоколів динамічної маршрутизації та протоколу SNMP, SNMP контролер та блок керування, які реалізують запропонований алгоритм балансування навантаження.

Список літератури

[1] Новиков О. П. Анализ эффективности методов маршрутизации видеоинформации в сетях интернет [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/n/analiz-effektivnosti-metodov-marshrutizatsii-videoinformatsii-v-setyah-internet.pdf.

[2] Ю. П. Лукашин, *Адаптивные методы краткосрочного прогнозирования*, М.: Статистика, 1979.

[3] В. К. Морозов, и А. В. Долганов, *Основы теории информационных сетей*, М.: Высшая школа, 1987.

[4] Ke Xu, Hongying Liu, Jiangchuan Liu, Jixiu Zhang, " LBMP: A Logarithm- Barrier-Based Multipath Protocol for Internet Traffic Management", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 3, MARCH 2011.

[5] Классификация методов многопутевой маршрутизации [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/n/klassifikatsiya-metodov-mnogoputevoy-marshrutizatsii.pdf.

[6] Sheng Huang, Biswanath Murkherejee. Adaptive Reliable Multi Path Provisioning in WDM Networks. / in Proc. of IEEE ICC — Beijing, China, 2008. Pp. 5300-5304.

[7] Lei Song, Biswanath Mukherjee.On the Study of Multiple Backups and Primary-Backup Link Sharing for Dynamic Service Provisioning in Survivable WDM Mesh Networks / IEEE Journal on selected areas in Telecommunication, 2008. Vol. 26, No6. pp. 84-91.

[8] А. Х. Панеш, «Достоинства и недостатки программно-конфигурируемых компьютерных сетей» *Вестник АГУ*. № 3(186), 2016 [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/n/dostoinstva-i-nedostatki-programmno-konfiguriruemyh-kompyuternyh-setey.pdf

[9] NGN: принципы построения и организации / под ред. Ю.Н. Чернышова. – М.: Эко-Трендз, 2008. – 400 с.

[10] Y.2001. ITU-T. Recommendation Y.2001: General overview of NGN, ITU-T. – Geneva, 2004. – 18 p.

[11] Вегенша Ш. *Качество обслуживания в сетях IP*. – М.: Издательский дом «Вильямс», 2003. – 368 с.

[12] Остерлох Х. *Маршрутизация в IP-сетях. Принципы, протоколы, настройка*, СПб.: BHV, 2002. – 512 с.

[13] Medhi D., Ramasamy K. Network routing: algorithms, protocols, and architectures. Morgan Kaufmann, 2007. – 788 p.

[14] Ю. А. Кулаков, А. В. Коган, В. М. Храпов, «Способ конструирования трафика при организации многопутевой маршрутизации,» *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка* – Вип. 65. – С. 28–33 2017. [Електронний ресурс]. – Режим доступу: http://ela.kpi.ua/handle/123456789/22143.

[15] Том М. Томас II. Структура и реализация сетей на основе протокола OSPF. Руководство Cisco = OSPF Network Design Solutions. — 2-е изд. — М.: «Вильямс», 2004. — с. 816

[16] М. П. Березко, В. М. Вишневский, Е. В. Левнер, Е. В. Федотов, «Математические модели исследования алгоритмов маршрутизации в сетях передачи данных,» *Информационные процессы. Электронный научный журнал*, Том 1, № 2, стр. 103-125, 2001.

Стаття надійшла: 24.08.2018.

Відомості про авторів

Захарченко Сергій Михайлович – к. т. н, доцент кафедри обчислювальної техніки, Вінницький національний технічний університет.
Шевчук Катерина Ігорівна – аспірантка кафедри обчислювальної техніки, Вінницького національного технічного університету.

S. M. Zakharchenko[1], K. I. Shevchuk[1]

METHOD FOR IMPROVEMENT OF DYNAMIC ROUTING PROTOCOLS

[1]Vinnytsia National Technical University

УДК 519.86(075.8)

В. В. Колодний[1], Д. С. Кудрявцев[1]

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВІЗУАЛЬНОГО МОДЕЛЮВАННЯ ТА ОБРОБКИ ТЕРНАРНИХ ГЕШТАЛЬТ-РАНЖУВАНЬ

[1]Вінницький національний технічний університет

Анотація. Розроблено нову інформаційну технологію, засновану на візуалізації тернарних гештальт-ранжувань та їхній подальшій комп'ютерній обробці. Під гештальт-ранжуванням розуміється одночасне пред'явлення особі, що приймає рішення (ОПР), декількох альтернатив для їхнього наочного порівняння з активізацією швидкої системи інтуїтивного мислення. Було зроблено висновок, що ОПР досить впевнено та надійно можуть одночасно оперувати трьома альтернативами і порівнювати їх на трьох рівнях, тобто здійснювати тернарні трирівневі гештальт-ранжування. Описано основні етапи та проаналізовано прикладні аспекти застосування даної інформаційної технології. Розроблена інформаційна технологія візуального моделювання та обробки тернарних гештальт-ранжувань не передбачає ніяких запитань до ОПР і не потребує від неї ніякої інформації в числовій або вербальній формі. Під час проведення кожного тернарного гештальт-ранжування візуалізується трійка альтернатив, які ОПР має розмістити на крузі переваг за принципом: чим вище якість альтернативи, тим ближче до центру круга переваг потрібно її розташовувати. Це збільшує ефективність використання швидкого інтуїтивного мислення ОПР із залученням візуальної інформації та зорової пам'яті. Продемонстровано роботу комп'ютерної програми та наочно представлено основні та додаткові можливості даної інформаційної технології візуального моделювання на прикладі вибору мови програмування для реалізації програмного продукту. Створену комп'ютерну програму можна використовувати як ефективний інструмент виявлення переважань для розв'язання багатьох практичних задач прийняття рішень з цільовою та критеріальною невизначеністю.

Ключові слова: інформаційна технологія, тернарне гештальт-ранжування, візуальне моделювання, виявлення переважань, критеріальна невизначеність.

Аннотация. Разработана новая информационная технология, основанная на визуализации тернарных гештальт-ранжирований и их дальнейшей компьютерной обработке. Под гештальт-ранжированием понимается одновременное предъявление лицу, принимающему решение (ЛПР), нескольких альтернатив для их наглядного сравнения с активизацией быстрой системы интуитивного мышления. Был сделан вывод, что ЛПР достаточно уверенно и надежно могут одновременно оперировать тремя альтернативами и сравнивать их на трёх уровнях, то есть осуществлять тернарные трёхуровневые гештальт-ранжирования. Описаны основные этапы и проанализированы прикладные аспекты применения данной информационной технологии. Разработанная информационная технология визуального моделирования и обработки тернарных гештальт-ранжирований не предусматривает никаких вопросов к ЛПР и не требует от нее никакой информации в числовой или вербальной форме. Во время проведения каждого тернарного гештальт-ранжирования визуализируется тройка альтернатив, которые ЛПР должно разместить на круге предпочтений по принципу: чем выше качество альтернативы, тем ближе к центру круга предпочтений нужно ее располагать. Это увеличивает эффективность использования быстрого интуитивного мышления ЛПР с привлечением визуальной информации и зрительной памяти. Продемонстрирована работа компьютерной программы и наглядно представлены основные и дополнительные возможности данной информационной технологии визуального моделирования на примере выбора языка программирования для реализации программного продукта. Созданную компьютерную программу можно использовать как эффективный инструмент выявления предпочтений для решения многих практических задач принятия решений с целевой и критериальной неопределённостью.

Ключевые слова: информационная технология, тернарное гештальт-ранжирование, визуальное моделирование, выявление предпочтений, критериальная непределённость.

Abstract. The new information technology based on the visualization of ternary gestalt-ranking and their subsequent computer processing has been developed. Under Gestalt-ranking means the simultaneous presentation for decision-maker of several alternatives for their visual comparison with the activation of the rapid system of intuitive thinking. It was concluded that decision-makers can confident and reliably operate at the same time with three alternatives and compare them at three levels, that is, to carry out ternary three-level gestalt rankings. The main stages are described and applied aspects of application of this information technology are analyzed. The information technology of visual modeling and processing of ternary gestalt-rankings is not designed to provide any questions to the decision-maker and does not require any information from him in numerical or verbal form. During each ternary-gestalt ranking, three alternatives are visualized, which decision-maker must place on the circle of advantages: the higher the quality of the alternative, the closer to the center of the circle of advantages it is necessary to position it. This increases the efficiency of using fast intuitive thinking of decision-maker with the use of visual information and visual memory. The work of the computer program is demonstrated and the basic and additional features of this information visual modeling technology are presented on an example of the choice of the programming language for the implementation of software product. The created computer program can be used as an effective detection of preferences tool for solving many practical tasks of decision making with target and criteria uncertainty.

Key words: information technology, ternary gestalt-ranking, visual modeling, detection of preferences, criteria uncertainty. DOI: https://doi.org/10.31649/1999-9941-2018-42-2-26-34.

Вступ

Кожна людина щоденно приймає багато різноманітних рішень, але більшість таких рішень відбувається інтуїтивно, на основі досвіду та здорового глузду. Існуючі системи підтримки прийняття рішень вимагають від ОПР навчання та певної фахової підготовки і спираються на запитання щодо переважань ОПР. Ці запитання потребують відповідей ОПР у вигляді значень числових або лінгвістичних змінних, що часто викликає труднощі. Оскільки переважна більшість практичних задач прийняття рішень є неструктурованими або слабо структурованими, коректність застосування для них багатьох існуючих строгих математичних методів викликає сумнів.

Актуальність

Слабо структуровані задачі прийняття рішень з цільовою та критеріальною невизначеністю зустрічаються на кожному кроці: це задачі конкурсного відбору, різноманітного рейтингування, експертного оцінювання, призначення на посаду тощо. Виявилося, що існуючі методи та інформаційні технології або взагалі не підходять для розв'язання таких задач прийняття рішень, або вони є занадто складними, трудомісткими та не наочними для ОПР. Тому завдання розробки нових надійних, наочних та простих в користуванні інформаційних технологій є актуальним.

Мета та задачі дослідження

Основною метою створення даної інформаційної технології було спрощення виконання ранжування у порівнянні з іншими методами та мінімізація когнітивних зусиль ОПР для підвищення надійності отримання результуючого ранжування альтернатив.

Ключовою задачею було створення принципіально нового інструменту визначення результуючого ранжування, базуючись на інтуїтивному сприйнятті та відносній простоті використання тернарних гештальт-ранжувань, що дає змогу значно збільшити цільову аудиторію та сфери застосування даної інформаційної технології. Додатковими задачами є адаптація під оптимальну для ОПР точність виявлення переважань, швидке виявлення можливих суперечливостей ОПР в реальному часі, демонстрація проміжних результатів, можливість збереження результатів для подальшої обробки, а також перегляд відносних переваг між альтернативами у візуальній формі.

Теоретичне підґрунтя застосування тернарних трирівневих гештальт-ранжувань для виявлення переваг ОПР

Опишемо теоретичні основи розробленої інформаційної технології візуального моделювання тернарних гештальт-ранжувань, що ґрунтуються на роботах [1-3].

Досить розповсюдженим є клас практичних задач прийняття рішень, що мають такі властивості:

- критерії для оцінювання та порівняння альтернатив частково або повністю невідомі (як самі критерії, так і їхні значення на певних шкалах);
- всі наявні альтернативи відомі і їхня загальна кількість є невеликою.

Такі задачі прийняття рішень будемо називати задачами для декількох альтернатив з критеріальною невизначеністю і в подальшому будемо розглядати саме цей клас задач, який, очевидно, є підкласом слабо структурованих задач прийняття рішень.

Інтуїтивно зрозуміло, що єдиним можливим способом структурувати множину альтернатив, тобто визначити переваги ОПР на цій множині, є пред'явлення ОПР для порівняння деяких з множини альтернатив. В найпростішому і тому найрозповсюдженому випадку альтернативи пред'являються двійками і здійснюється попарне порівняння всіх альтернатив. Якщо ОПР фіксує лише факт переваги однієї альтернативи над іншою, достатньо двох відношень та двох рівнів порівняння (рис. 1).

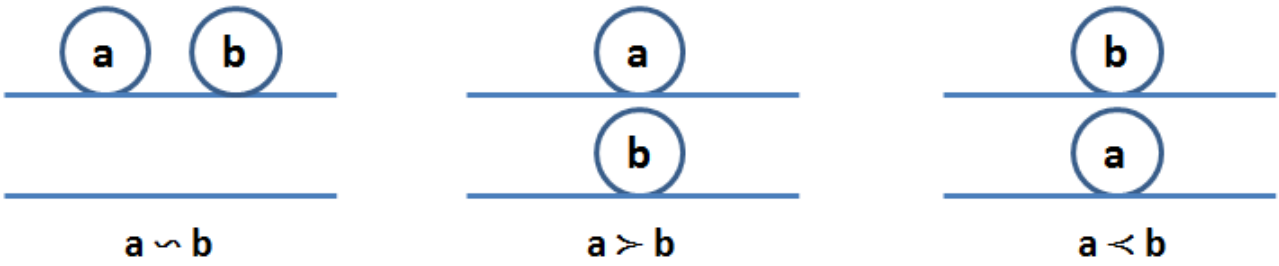


Рисунок 1 – бінарні ранжування на двох рівнях (попарні порівняння)

Якщо ОПР може визначити не лише факт, але і ступінь переваги однієї альтернативи над іншою, застосовується більша кількість рівнів бінарних порівнянь. Наприклад, в шкалі Сааті попарні порівняння відбуваються на дев'яти рівнях [4].

Багаторічне практичне застосування цієї шкали продемонструвало, що 9 рівнів буває забагато для надійного оцінювання ступеня переваги альтернатив ОПР, що не є професійними системними аналітиками. З іншого боку, досвідчені фахівці-професіонали, що мають розвинене числове мислення, іноді вважають, що 9 рівнів шкали недостатньо для адекватного вимірювання ступеня переваг. Саме тому постає задача «підлаштовування» під ОПР і надання їй можливості працювати з найбільш прийнятною кількістю рівнів порівнянь альтернатив в кожній конкретній ситуації прийняття рішень. Навіть Сааті визнає певну штучність своєї шкали і можливість отримання ненадійних оцінок від ОПР [4]. Розглянемо, наприклад, вербальну інтерпретацію оцінки «6» за дев'ятибальною шкалою Сааті: «краще, ніж істотно краще,

але гірше, ніж значно краще». Зрозуміло, що така словесна конструкція є громіздкою і навряд чи може зустрітися в природній мові.

Практика показує, що в більшості випадків двох рівнів порівняння (відношень $\succ$ і $\backsim$) виявляється явно замало. ОПР досить часто намагається не обмежуватися лише констатацією факту переваги однієї альтернативи над іншою, але й спробувати хоч якось оцінити ступінь цієї переваги (причому, як правило, вербально, а не кількісно!).

Прикладами таких розповсюджених вербальних оцінок ОПР є словесні конструкції:

(1) «альтернатива **A** значно переважає альтернативу **B**»;

(2) «альтернатива **B** безумовно краща ніж **A**, але трохи гірша, ніж **C**»;

(3) «альтернатива **A** набагато краща за **B**, але приблизно однакова за якістю з **C**».

В результаті проведених досліджень було зроблено висновок, що в більшості випадків ОПР можуть і бажають визначити ступінь переваги між альтернативами, але впевнено розрізняють лише два ступеня переваг: звичайну строгу перевагу $\succ$ та сильну (безумовну, беззаперечну) перевагу $\succ\succ$ [2, 3]. З урахуванням відношення еквівалентності ($\backsim$) маємо три відношення та три рівні порівняння для двох альтернатив (рис. 2).

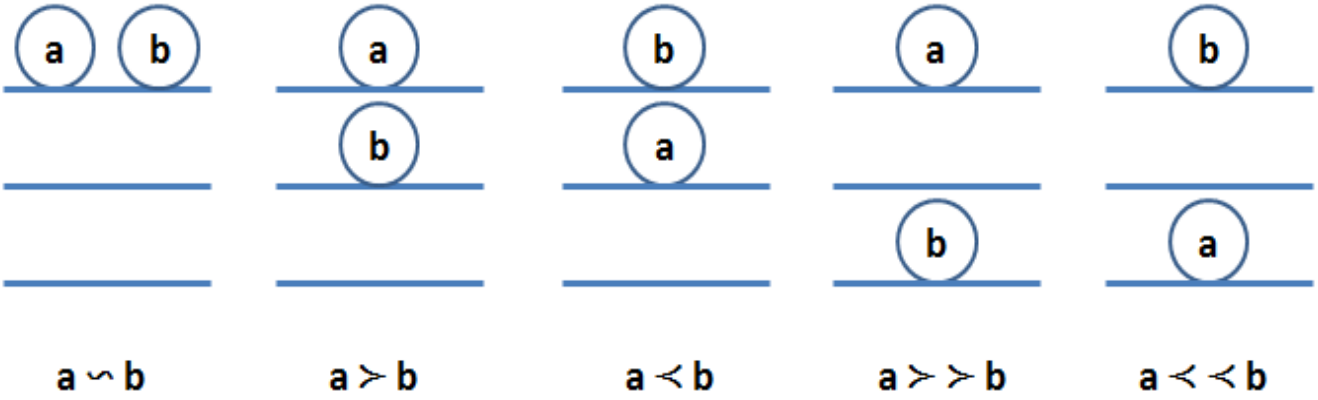


Рисунок 2 – бінарні трирівневі ранжування

Також виявилось, що ОПР досить часто розширюють контекст порівнянь, намагаючись додати до попарного порівняння третю альтернативу, як в словесних конструкціях (2), (3). Зрозуміло, що попарних порівнянь в таких випадках для адекватного відображення думки ОПР виявляється недостатньо.

З урахуванням введення символу сильної переваги можна досить легко та наочно записати в символьній формі словесні конструкції (1) – (3):

$$(1) \Leftrightarrow A \succ\succ B;$$

$$(2) \Leftrightarrow C \succ B \succ\succ A;$$

$$(3) \Leftrightarrow A \backsim C \succ\succ B.$$

Подальше розширення контексту порівнянь ОПР, як правило, не відбувається, тому що згадування в одному реченні чотирьох і більшої кількості альтернатив з описом ступеня переваг між ними робить відповідну словесну конструкцію дуже громіздкою та мало зрозумілою. Тому було зроблено такий висновок: **ОПР досить впевнено та надійно можуть одночасно оперувати трьома альтернативами і порівнювати їх на трьох рівнях, тобто здійснювати тернарні трирівневі гештальт-ранжування.**

Поняття гештальт-ранжування введено в роботі [3] і означає одночасне пред'явлення ОПР декількох альтернатив для їхнього наочного порівняння з активізацією швидкої системи інтуїтивного мислення ОПР [5].

З усіх гештальт-ранжувань найбільш природними, надійними та ефективними є саме тернарні трирівневі гештальт-ранжування [3]. Кожне тернарне трирівневе гештальт-ранжування породжує трійку бінарних трирівневих ранжувань.

Очевидно, що тернарні трирівневі гештальт-ранжування, зображені на рисунку 3, надають ОПР набагато більші можливості різноманітно, надійно та точно виразити свої переваги між різними альтернативами, ніж бінарні порівняння на двох (рис. 1) та трьох (рис. 2) рівнях. Зрозуміло, що роздільна здатність тернарних трирівневих ранжувань є набагато більшою.

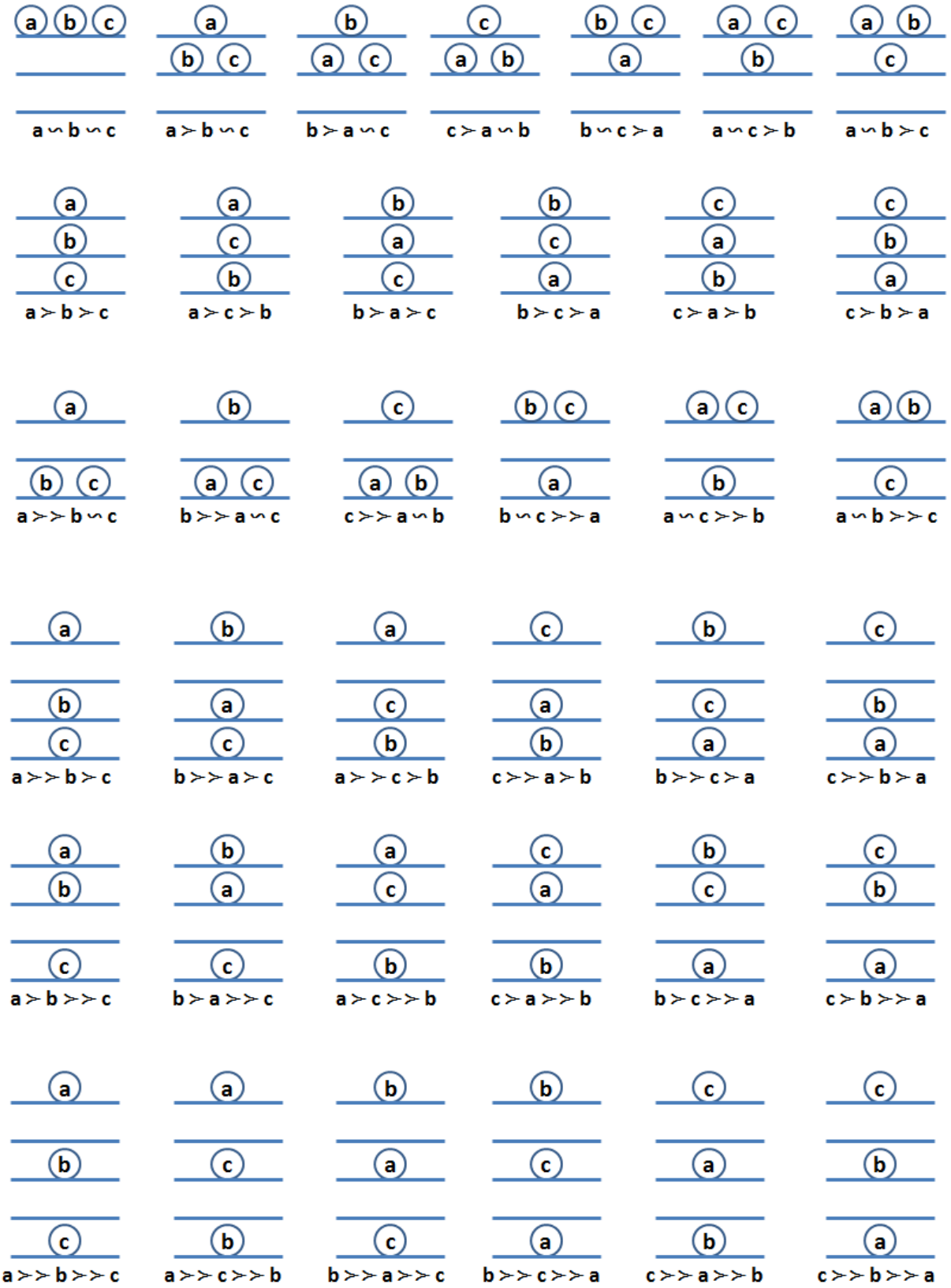


Рисунок 3 – тернарні трирівневі гештальт-ранжування

Візуальні та відповідні символьні моделі тернарних трирівневих ранжувань на рисунку 3 є досить наочними та зрозумілими для системного аналітика, але, на жаль, вони виявилися незручними для ОПР, які не є фахівцями в галузі теорії прийняття рішень. Тому було поставлено завдання створити нову зруч-

ну та інтуїтивно зрозумілу інформаційну технологію для візуалізації та подальшої комп’ютерної обробки інформації у вигляді тернарних гештальт-ранжувань.

Опис розробленої інформаційної технології

Розроблена інформаційна технологія візуального моделювання та обробки тернарних гештальт-ранжувань не передбачає ніяких запитань до ОПР і не потребує ніякої інформації від ОПР в числовій або вербальній формі. Потрібно лише перетягнути зображення трійки кружечків-альтернатив на круг переваг згідно з вподобаннями ОПР. Це збільшує ефективність використання швидкого інтуїтивного мислення [5] із залученням візуальної інформації та зорової пам’яті ОПР. Таким чином, під час проведення кожного тернарного гештальт-ранжування візуалізується трійка альтернатив, які ОПР має розмістити на крузі переваг за принципом: чим вище для ОПР якість альтернативи, тим ближче до центру круга переваг потрібно її розташовувати.

Розроблена інформаційна технологія складається з таких етапів:

1. Введення бажаних для ОПР кількості і назв альтернатив та кількості градацій (рівнів) порівняння. Кількість градацій ОПР може змінювати за своїм бажанням в процесі роботи з програмою.
2. Проведення тернарних гештальт-ранжувань з візуальним моделюванням на крузі переваг.
3. Числове вимірювання якості альтернатив з урахуванням їхнього розташування на крузі переваг та автоматична генерація (отримання) трійки бінарних трирівневих ранжувань, що відповідають поточному тернарному гештальт-ранжуванню.
4. Аналіз кожного отриманого поточного бінарного трирівневого ранжування на суперечливість з раніше отриманими бінарними трирівневими ранжуваннями.
5. Призупинення гештальт-ранжувань з можливістю пошуку та виправлення виявлених помилок ОПР.
6. Візуалізація всіх проведених тернарних гештальт-ранжувань, в які входить задана альтернатива.
7. Повернення до попередніх гештальт-ранжувань з метою виправлення виявлених помилок.
8. Обчислення результуючих нормованого та центрованого кардинальних ранжувань всієї множини альтернатив.
9. Візуалізація спектру переваг усіх альтернатив та обчислених результуючих ранжувань у вигляді кольорових діаграм.

Інформаційна технологія візуального моделювання тернарних гештальт-ранжувань на крузі переваг реалізована у вигляді комп’ютерної програми, користувачами якої можуть бути як системні аналітики та фахівці з теорії прийняття рішень, так і ОПР-непрофесіонали. Для зручності використання даної технології розроблено наочний навігаційний інтерфейс із наявністю інтерактивної системи підказок та повідомлень при некоректних діях користувача.

На рисунку 4 зображено схему переходів між основними вікнами програми.

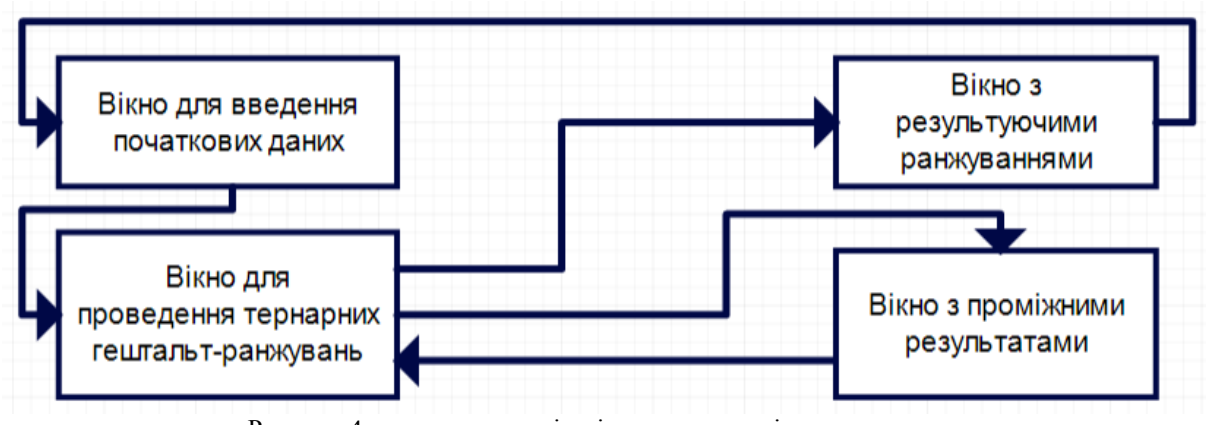


Рисунок 4 – схема переходів між основними вікнами програми

Користувач має змогу обрати кількість альтернатив (від 4 до 9 включно) та ввести їхні назви. Кількість рівнів для порівняння альтернатив (градацій) може обиратися користувачем та варіюється від 5 до 500. Вибір кількості градацій вперше здійснюється у вікні для введення початкових даних, а її зміна може здійснюватись на будь-якому кроці проведення тернарних гештальт-ранжувань.

Розглянемо застосування розробленої інформаційної технології для розв’язання задачі прийняття рішень в умовах критеріальної невизначеності на прикладі вибору мови програмування для реалізації програмного продукту із залученням кваліфікованого експерта. Спочатку необхідно виконати налашту-

вання (рис. 5), тобто обрати кількість альтернатив (шість) та кількість градацій якості для їхнього порівняння (десять). Після цього можна ввести назви альтернатив (мов програмування, що порівнюються).

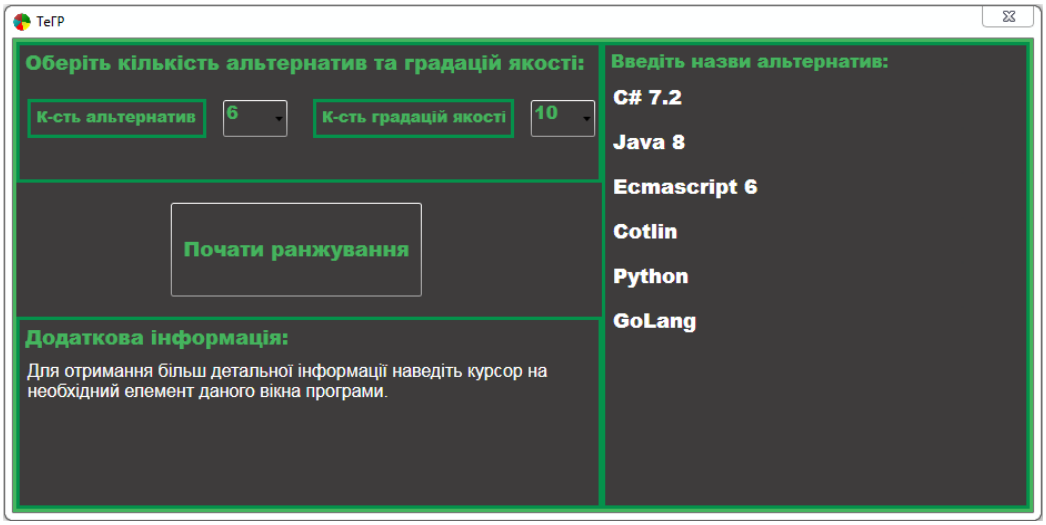


Рисунок 5 – вікно для введення початкових даних

На рисунку 6 зображено вікно програми для проведення тернарних гештальт-ранжувань.

Рисунок 6 – вікно програми для проведення тернарних гештальт-ранжувань

Якщо під час проведення поточного гештальт-ранжування виникає суперечливість з раніше проведеними, вона буде негайно виявлена (рис. 7).

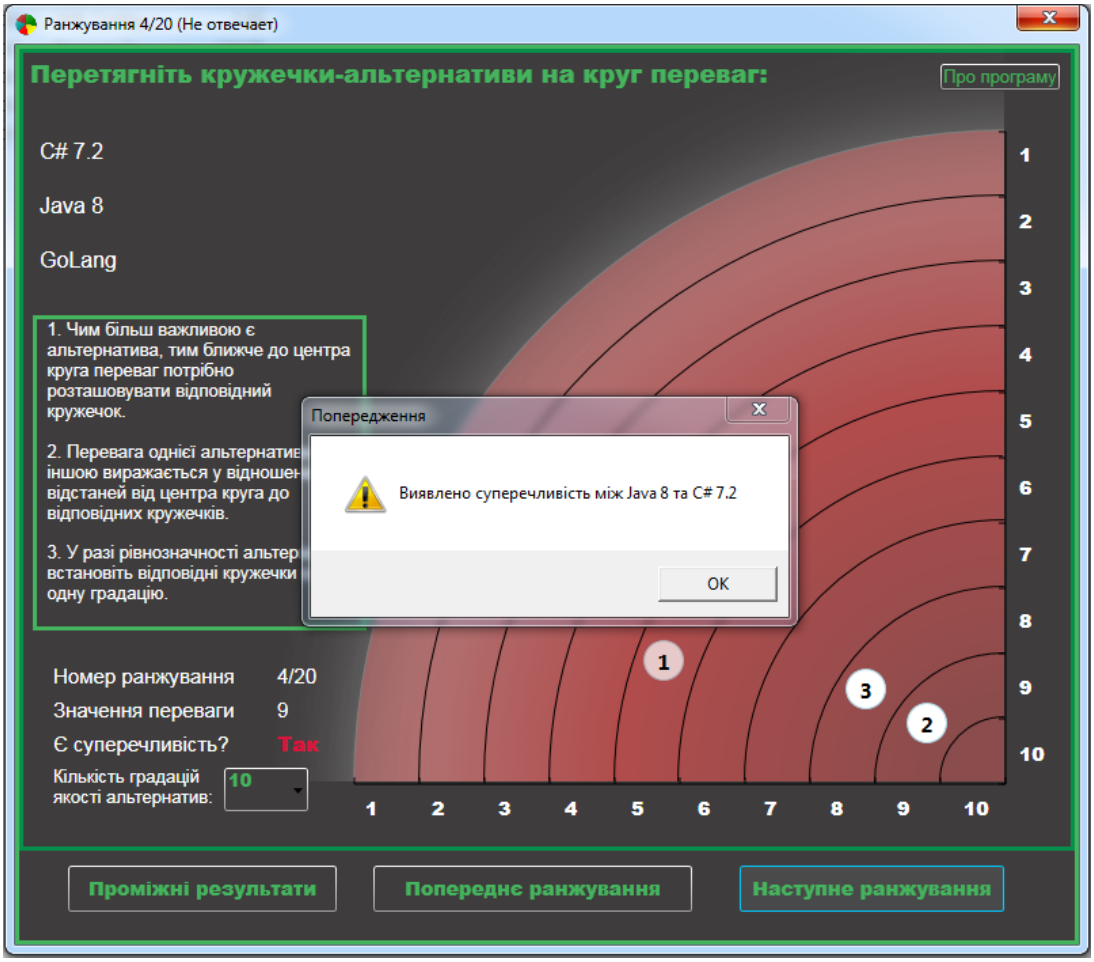


Рисунок 7 – повідомлення про виявлену суперечливість

Користувач має змогу не лише отримувати повідомлення щодо виявлених програмою суперечливостей, але й наочно бачити суперечливість, використовуючи кругові діаграми в проміжних результатах (рис. 8).

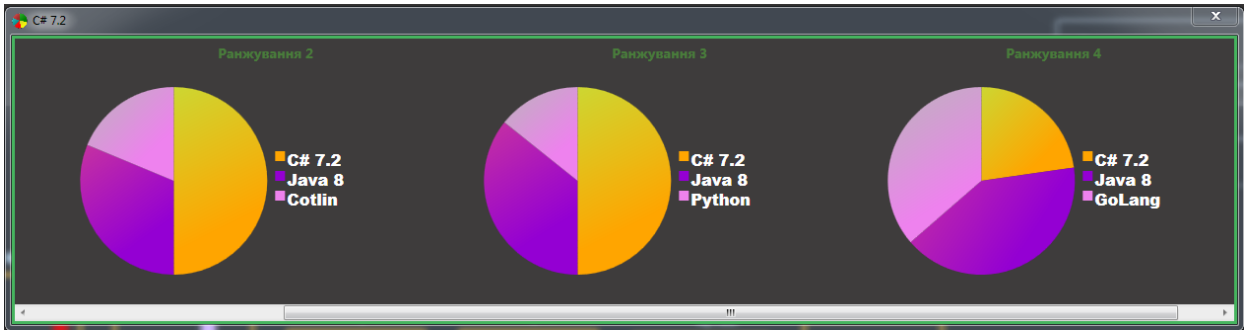


Рисунок 8 – Графічне представлення суперечливості між альтернативами Java 8 та C# 7.2

Результати виконання програми (рис. 9) містять нормоване та центроване результуючі кардинальні ранжування усієї множини альтернатив, а також графічні моделі (спектри переваг) альтернатив, представлені стовпчиковими діаграмами, що показують кількість породжених бінарних трирівневих ранжувань, які були отримані під час проведення тернарних гештальт-ранжувань. Графічні моделі альтернатив є інтуїтивно зрозумілими для користувача та виділяються різними кольорами, що відповідають відношенням між альтернативами.

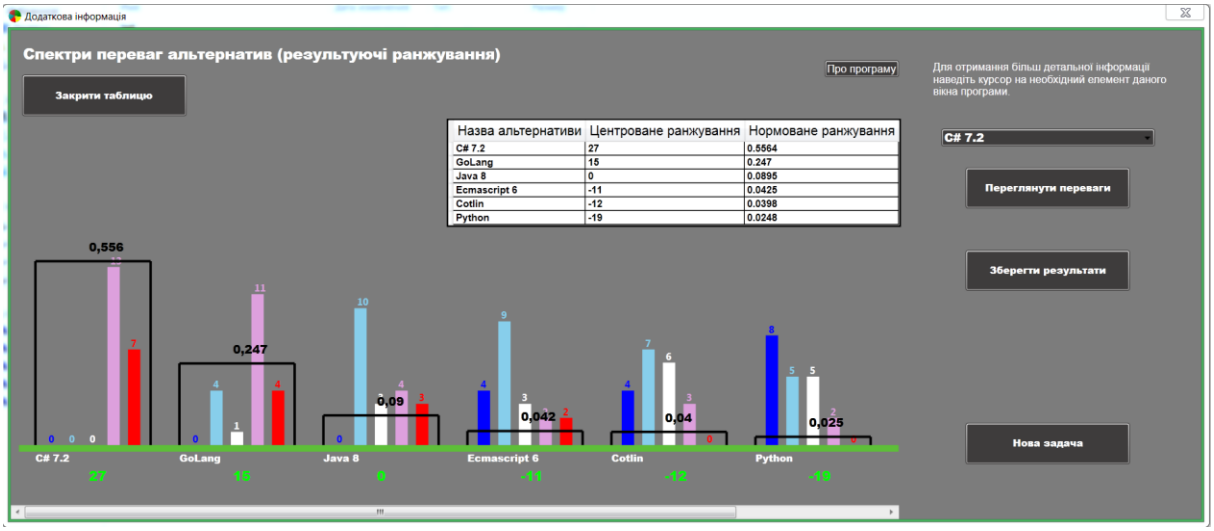


Рисунок 9 – вікно з результуючими ранжуваннями

Аналізуючи отримані результати для нашого прикладу (рис. 9), можна зробити висновок, що експерт вважає найбільш підходящою мовою програмування C# 7.2. Мова GoLang також є підходящою, а мови Ecmascript 6, Cotlin, Python зовсім не підходять для створення даного програмного продукту.

Користувачеві надана можливість збереження результатів у вигляді фото графічних моделей альтернатив та текстового файлу із значеннями абсолютних переваг альтернатив для кожного проведеного тернарного гештальт-ранжування.

Розроблену інформаційну технологію було реалізовано у вигляді комп'ютерної програми «Інформаційна технологія візуального моделювання тернарних гештальт-ранжувань на крузі переваг (ТеГР)» у середовищі програмування Visual Studio 2015 на мові C# із використанням технології WPF та пакету бібліотек платформи .NET Framework 4.6 [6].

Висновки

1. Розроблена інформаційна технологія виявлення переваг є зручною для ОПР, наочною, надійною та не складною у використанні.

2. Альтернативи і переваги між ними представлено у вигляді простих візуальних моделей, що потребує від ОПР значно менших когнітивних зусиль порівняно з традиційними методами, заснованими на отриманні значень числових або лінгвістичних змінних.

3. Створену на основі цієї інформаційної технології комп'ютерну програму можна використовувати як ефективний інструмент виявлення переважань для розв'язання багатьох практичних задач прийняття рішень з цільовою та критеріальною невизначеністю.

Список літератури

[1] В. В. Колодний, «Трирівневі ранжування та їх застосування для виявлення переважань,» на *Контроль і управління в складних системах*, Вінниця, 2003, с. 238.

[2] В. В. Колодний, та В. В. Зубко, «Метод некритеріального структурування множини альтернатив за допомогою аналізу тернарних трирівневих ранжувань,» на *ІНТЕРНЕТ–ОСВІТА–НАУКА–2014*: Вінниця, 2014, с. 13-14.

[3] В. В. Колодний, та В. В. Зубко, «Застосування гештальт-ранжувань для виявлення переваг ОПР,» на *ІНТЕРНЕТ-ОСВІТА-НАУКА-2016*, Вінниця, 2016, с. 43-44.

[4] Т. Л. Саати, *Принятие решений. Метод анализа иерархий* Москва: Радио и связь, 1989.

[5] Д. Канеман, Думай медленно… решай быстро Москва: АСТ, 2014.

[6] Д. С. Кудрявцев, та В. В. Колодний, «Комп'ютерна програма "Інформаційна технологія візуального моделювання тернарних гештальт-ранжувань на крузі переваг» (ТеГР)" *Свідоцтво про реєстрацію авторського права на твір* №78345. Дата реєстрації 17.04.2018.

Стаття надійшла: 28.08.18.

Відомості про авторів

Колодний Володимир Володимирович – кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук.

Кудрявцев Дмитро Станіславович – студент кафедри комп'ютерних наук, факультету інформаційних технологій та комп'ютерної інженерії.

V. V. Kolodnyi[1], D. S. Kudryavtsev[1]

INFORMATION TECHNOLOGY OF VISUAL MODELING AND PROCESSING TERNARY GESCHTALT-RANKINGS

[1]Vinnytsia National Technical University

УДК 004.42:519.873

Р. В. Крепич[1], С. Я. Крепич[1]

КОМПЛЕКСНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЗБОРУ І ВІЗУАЛІЗАЦІЇ СТАТИСТИКИ ДЕФЕКТІВ ПРОГРАМНИХ ПРОЕКТІВ

[1]Тернопільський національний економічний університет

Анотація. В статті розглянуті вимоги до даних для побудови моделі надійності програмної системи. За допомогою UML діаграм класів описані процеси збору, трансформації (переведення з одного формату даних в інший), аналізу і візуального представлення інформації, щодо кількості зареєстрованих на проектах дефектів. Розроблений програмний комплекс, на основі поставлених вимог, надасть можливість проводити подальші дослідження моделювання надійності програмних систем з урахуванням кількості дефектів і часу їх реєстрації у відповідності до тривалості проекту. Планується провести детальний аналіз зібраних даних задля визначення закономірностей залежності кількості дефектів від різних характеристик проекту (складність архітектури, тривалість проекту, кількість тестувальників, кількість розробників). Більшість моделей надійності програмних систем використовують обмежену (не повну) кількість характеристик проекту, які впливають на надійність програмного продукту, тому планується проаналізувати існуючи моделі надійності програмних продуктів і перевірити їх на зібраних даних задля того, щоб визначити додаткові характеристики проекту, що потрібно враховувати для моделювання надійності програмного продукту.

Ключові слова: дефект, модель надійності програмного забезпечення, проект, база даних, густина дефекту.

Аннотация. В статье рассмотрены требования к данным для построения модели надежности программной системы. С помощью UML диаграмм классов описаны процессы сбора, анализа и визуального представления информации, по количеству зарегистрированных на проектах дефектов. Разработанный программный комплекс на основе поставленных требований, позволит проводить дальнейшие исследования моделирования надежности программных систем с учетом количества дефектов и времени их регистрации в соответствии с продолжительностью проект. Планируется провести детальный анализ собранных данных для определения закономерностей зависимости количества дефектов от различных характеристик проекта (сложность архитектуры, продолжительность проекта, количество тестировщиков, количество разработчиков). Большинство моделей надежности программных систем используют ограниченную (не полную) количество характеристик проекта, которые влияют на надежность программного продукта, поэтому планируется проанализировать существующие модели надежности программных продуктов и проверить их на собранных данных для того, чтобы определить дополнительные характеристики проекта, потриб- но учитывать для моделирования надежности программного продукта

Ключевые слова: дефект, модель надежности программного обеспечения, проект, база данных, плотность дефекта

Abstract. In the article the data requirements for construction of the reliability model of the software system are considered. Using UML class diagrams, we describe the processes of collecting, analyzing and visual representation of information on the number of defects registered on projects. The developed software package, based on the requirements, will provide the opportunity to conduct further research on the simulation of the reliability of software systems, taking into account the number of defects and the time of their registration in accordance with the duration of the project. It is planned to carry out a detailed analysis of the collected data in order to determine the patterns of dependence of the number of defects on the various project characteristics (complexity of the architecture, duration of the project, number of testers, number of developers). Most software system reliability models use a limited (not complete) number of project characteristics that affect the reliability of the software, so it is planned to analyze the existing software product reliability models and verify them on the collected data in order to determine the additional characteristics of the project, which is needed, but take into account for modeling the reliability of a software product

Key words: defect, software reliability model, project, database, defect densitydefect.
DOI: https://doi.org/10.31649/1999-9941-2018-42-2-35-42.

Вступ

На сьогоднішній день актуальною є проблема дослідження надійності програмної системи (ПС)[1]. Під надійністю програмної системи розуміємо здатність системи виконувати покладені на неї функції, протягом певного часу експлуатації в наперед визначених допустимих межах та за певних умов експлуатації[2-4]. Для багатьох систем надійність є основною характеристикою і критичною вимогою (системи реального часу, радарні системи, системи безпеки, медичне устаткування з вбудованими програмами та ін)[4-6], оскільки відмова такої системи або її неправильне функціонування призводить до серйозних наслідків, а час, необхідний для усунення дефектів системи у процесі її експлуатації, значно довший ніж під час перевірки програмної системи (етап тестування)[6].

Огляд особливостей математичних моделей надійності програмних систем[7-10] показав їх обмежене застосування, оскільки по своїй суті вони є статичними і відображають зв'язки між загальною кількістю кумулятивних дефектів у програмній системі. Взамін зазначеного класу моделей пропонується використовувати моделі динаміки, кумулятивних похибок у процесі проведення тестування, зокрема на стадії регресійного та інтегративного тестування. Однак перед тим, як перейти до моделювання надійності програмних систем за обраною схемою потрібно провести додатковий збір, аналіз і візуальне представлення реальних даних з загально доступних джерел, що є метою даної статті. Важливо провести збір даних з якомога більшої кількості різних джерел, щоб отримати якомога більшу можливість для аналізу, покращення існуючих моделей надійності ПС на основі кумулятивних помилок.

Мета дослідження

Розробка програмного забезпечення для збору даних з різних джерел, перевід отриманих даних в єдиний формат для подальшої візуалізації і аналізу на існуючих методах моделювання надійності програмних систем. Проведений аналіз дозволить вирахувати на основі великої кількості різноманітних даних додаткові коефіцієнти невраховані попередньо у загальновідомих моделях і тим самим підвищити їхню якість.

Постановка задачі

Основною метою збору і аналізу даних про дефекти проектів є отримання достатньої кількості інформації для розгляду і удосконалення моделей надійності програмного забезпечення, що відносяться до моделей кумулятивних помилок. Дані моделі використовуються для дослідження функціональної залежності інтенсивності відмов на помилку та інтенсивності відмов протягом певного періоду часу. Оскільки, здебільшого користувачеві не надається інформація про внутрішню структуру програмного продукту і висновки про його надійність можна зробити лише оцінивши його вхідні та вихідні дані, які можна використати для параметричної і структурної ідентифікації відомих моделей надійності програмних систем [7-10]. Тому у роботі доцільно розглядати моделі «чорної скриньки», предметом дослідження яких є кількість помилок у визначеному часовому інтервалі.

Найбільш поширеними моделями цього типу є деєутрофікаційна модель Jelinski та Moranda, модель Schick та Wolverton, геометрична деєутрофікаційна модель Jelinski та Moranda, геометрична пуассонова модель Moranda, модель недосконалого відлагодження Goel та Okumoto та ін[11].

Усі ці моделі базуються на наступних основних припущеннях [12]:

- Кумулятивна кількість відмов на момент t, $M(t)$ відповідає пуассоновому процесу з функцією математичного сподівання $\mu(t)$. При цьому, очікувана кількість помилок, які буде виявлено за проміжок часу $t + \Delta t$ пропорційна кількості помилок, що залишилися у системі на момент t. Також вважається, що $\mu(t)$ - обмежена, не спадаюча функція часу: $lim_{t\to\infty}\mu(t) = N < \infty$. Отже, ця модель належить до категорії скінчених функцій.
- Кількості помилок $f_1, f_2, \dots, f_n$, виявлених на відповідних часових інтервалах $[(t_0 = 0, t_1), (t_1, t_2), \dots, (t_{n-1}, t_n)]$ незалежні для будь якої скінченої кількості часових проміжків $t_1 < t_2 < \cdots < t_n$.
- Імовірності виявлення будь якої помилки є однаковими.
- При виявленні помилки, з програмного коду видаляється тільки один дефект без уведення нових.

При побудові моделі оцінювання та прогнозування надійності ПЗ кількість виявлених помилок розподілена за законом Пуассона. Динамічний показник складності проекту є параметром моделі та визначається на основі експериментальних даних і набуває значення додатного числа.

Функція інтенсивності виявлення помилок ПЗ прийме вигляд[12]:

$$\lambda(t) = \alpha\beta^{n+1}t^n\exp(-\beta t) \quad (1)$$

де α - коефіцієнт, який характеризує загальну кількість помилок, які були виявлені в програмі від початку спостереження, β - коефіцієнт, що характеризує швидкість зміни функції інтенсивності виявлення помилок, (завжди $\beta > 0$), n – параметр для оцінки величини проекту, де вибір параметру n залежить від процесу проведення тестування.

Для виразу (1) функція кумулятивної кількості помилок ПЗ має вигляд[12]:

$$\mu(t) = \int_0^t \lambda(\tau)d\tau = \alpha\left(n! - \sum_{i=0}^{n}\frac{n!\beta^{n-i}}{(n-1)!}t^{n-i}\,e^{-\beta t}\right) \quad (2)$$

Вирази (1) та (2) є моделлю з динамічним показником складності проекту.

Дана модель була вибрана для подальшого дослідження і покращення. З огляду на те, що на надійність ПС впливає багато чинників, такі як: термін, вартість, розмір, складність проекту, методологія його розробки і інші чинники, обґрунтованою є потреба перевірити існуючу модель надійності ПС на великій кількості даних різних проектів. Дана перевірка дасть можливість виявити, для яких проектів вибрана модель не є повноцінною і чому, тобто вона дасть можливість подальших досліджень додаткових параметрів, що варто враховувати при прогнозуванні надійності ПС, зав'язків між ними і їхню вагу в моделі.

Однак, для збору великої кількості даних потрібно розробити програмний комплекс, що спрямований на збір даних з різних ресурсів з різним форматом і їх представлення в єдиний формат для подальшого аналізу. Для збору даних було обрано загальнодоступні ресурси [13-16]

Тобто, поставленою задачею є розробка ПЗ, що дасть можливість збирати дані різного формату з багатьох джерел, подальше їх перетворення в єдиний формат для застосування в існуючій моделі, що дасть можливість проаналізувати існуючі параметри і коефіцієнти даної моделі, від чого вони залежать, а

також виявити додаткові параметри і коефіцієнти, введення яких в модель дасть можливість її покращення для більш точного прогнозування надійності ПС.

Вимоги до даних для побудови моделі

Одною з основних складових для побудови моделей є наявність достатньої кількості емпіричних даних, які в свою чергу, мають бути коректними і адекватними, в іншому разі буде не можливо побудувати потрібну модель, яка б змогла найточніше відтворювати реальні процеси і взаємодію даних. Тобто, одним з перших етапів побудови моделей є збір вхідних і вихідних даних, причому, кількість даних має бути якомога більшою, щоб включити якомога ширший спектр можливих сценаріїв внутрішніх процесів моделі.

В даному випадку, нас цікавить наступна інформація:

- проект і його основні атрибути (опис, основна ціль, початок і, можливо, завершення);
- інформація про дефекти проектів, що в свою чергу складається з
  - важливості дефекту, що означає на скільки дефект впливає на систему в загальному (для прикладу, якщо дефект спричиняє падіння системи, то його важливість критична). Дана інформація нам потрібна, щоб мати можливість працювати з дефектами в розрізі їхньої важливості і впливу на програмний продукт. Потрібно зазначити, що не завжди дефекти з високою важливістю будуть опрацьовані командою розробників в першу чергу. Тут більш важливий пріоритет дефекту, зокрема, якщо дефект має критичну важливість, але відтворюється вкрай складно або рідко, то він може отримати пріоритет меншого рівня і бути опрацьованим пізніше;
  - пріоритету дефекту, що означає на скільки дефект важливо виправити в плані часу. Тобто якщо дефект має високий пріоритет, то його потрібно опрацювати в першу чергу, бо він спричиняє некоректне функціонування важливих частин продукту;
  - дати реєстрації (тобто, коли дефект був знайдений і зареєстрований), щоб мати можливість проаналізувати частоту появи дефектів в часі (кількість знайдених дефектів на одиницю часу);
  - дата виправлення, щоб мати інформацію про час, який був потрібним для виправлення даного дефекту.
- більш детальна інформація про дефекти, що включає:
  - ким даний дефект був виявлений;
  - до якого модуля належить даний дефект;
  - детальний опис з кроками для відтворення;
  - людина, що має виправити даний дефект.

Оскільки для побудови точнішої моделі потрібні більш глибші (більша кількість однорідної інформації) та більш ширші (більша кількість атрибутів одної одиниці інформації) дані, доцільно розробити програмний продукту, що надасть можливість:

- працювати з декількома зовнішніми системами, що містять необхідну інформацію;
- отримувати з зовнішніх систем і тимчасово зберігати дані потрібної глибини і ширини в довільному форматі (формат даних відрізняється в залежності від системи, з якої їх отримано), який дасть змогу швидше і якісніше проаналізувати структуру даних;
- зберігати дані в кінцевому форматі в базу даних для подальшої зручної обробки;
- представляти збережені дані в різних форматах, для візуального аналізу, зокрема:
  - у вигляді таблиць з основною і додатковою інформацією;

у вигляді графіків різного спрямування (по часовому періоду і по типу графіку).

Вимоги до програмного забезпечення

З огляду на специфіку використання розроблюваного програмного забезпечення, поставлено наступні вимоги:

- підтримка декількох зовнішніх систем збору даних про дефекти проектів;
- можливість швидко збору, опрацювання і збереження даних;
- «портативність» - можливість використання програмного забезпечення на різних комп'ютерних станціях на основі операційної системи Windows без потреби встановлення додаткового програмного забезпечення. Щоб задовільнити дану вимогу пропонується використовувати базу даних SQLite, що не потребує встановлення серверної компоненти;
- зручний і зрозумілий інтерфейс з підтримкою української і англійської мови;
- наявність графічного відображення даних в декількох режимах;
- можливість налаштування поведінки програмного комплексу – зміни початкових налаштувань системи;
- сторінковий перегляд даних в табличному режимі – оскільки передбачається велика кількість даних, необхідною є умова можливості перегляду даних порціями;

- зберігання останніх вибраних змін користувача в базу даних (для зменшення кількості необхідних кроків користувача для отримання цікавої для нього інформації), зокрема:
    - збереження останнього вибраного користувачем проекту;
    - збереження вибраного часового режиму графіка (денний, тижневий, місячний);
    - збереження вибраного типу графіка (лінійний чи кумулятивний).

Спираючись на вище описане, визначимо, які основні функції програмного продукту будуть доступні користувачеві за допомогою діаграми варіантів використання представленої на рисунку 1.

Як видно з рисунку, користувач має можливість отримати дані про проекти і їх дефекти з різних джерел з можливістю збереження в базу даних та подальшим аналізом, який включає в себе візуалізацію даних в декількох представленнях

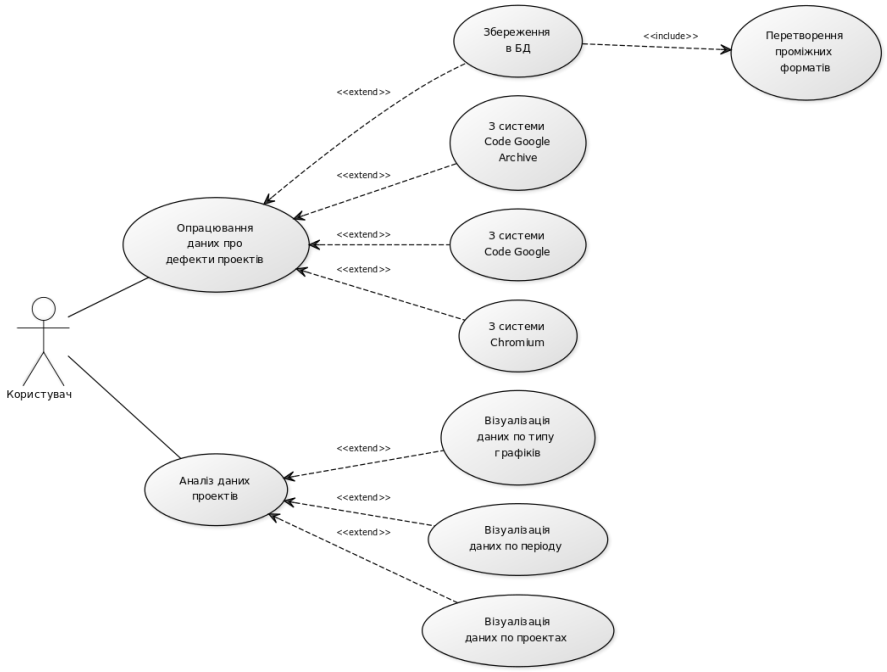


Рисунок 1 - Діаграма варіантів використання розроблюваного програмного комплексу

Архітектура програмного забезпечення

Коротко розглянемо структуру модуля отримання даних. На рисунку 2 наведена діаграма компонентів модуля отримання даних з різних систем і переведення їх в єдиний формат. Оскільки дані в кожній системі представлені в різному вигляді, потрібно розробити компонент, що буде розпізнавати потрібні дані для кожної системи. Даний компонент розпізнавши потрібні дані зберігає їх в проміжному форматі, зручному для аналізу і приведення до єдиного формату [17].

Рисунок 2 - Діаграма компонентів модуля отримання даних і переводу в єдиний формат

На рисунку 3 наведена діаграма основних класів, які використовуються для отримання даних з зовнішніх систем. Оскільки кожна система є унікальною, тому для кожної системи розроблений окремий клас, що містить всі необхідні методи для отримання даних. Дані класи наслідуються від базового класу, який містить загальні методи і властивості для всіх систем

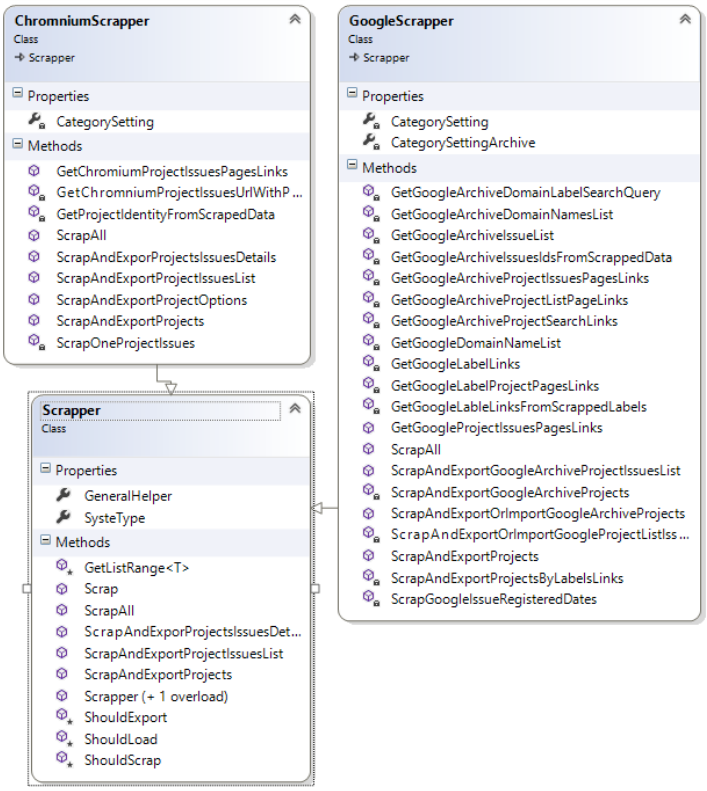


Рисунок 3 - UML діаграма основних класів модуля отримання даних

Приклад застосування

Розглянемо один із можливих сценаріїв роботи користувача з розробленим програмним продуктом збору, аналізу і візуалізації даних.

На першому етапі користувачеві потрібно отримати дані проекту, який його цікавить. Для цього він може скористатися меню програмного комплексу, що дає йому можливість отримати і імпортувати дані про проект і його дефекти в базу даних (див.рис.4)

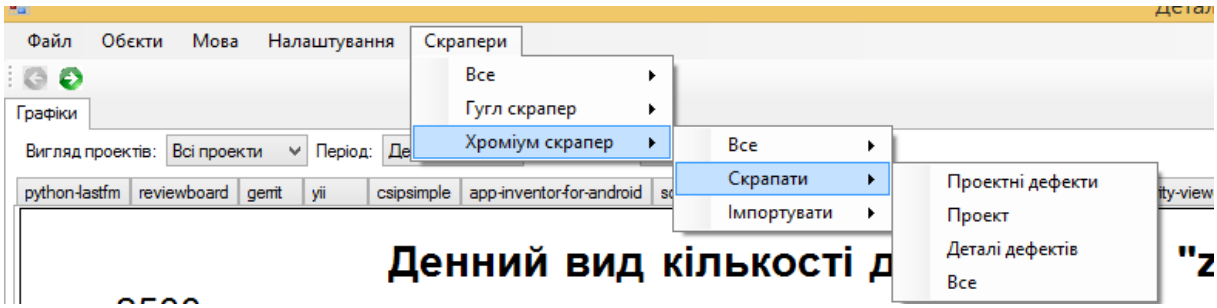


Рисунок 4 - Інтерфейс представлення можливості отримання/імпортування даних

Отримавши дані з зовнішної системи, користувач має можливість переглянути їх в табличному вигляді, який є зручним, коли потрібно переглянути деталі отриманих даних. На рисунку 5 представлене табличне відображення даних.

Також користувач має можливість переглянути дані про дефекти проекту у вигляді лінійного графіку. На рисунку 6 представлене графічне відображення інформації про кількість дефектів виявлених в проекті до поточного моменту часу, що дає користувачеві можливість відслідкувати пікові значення кількості дефектів.

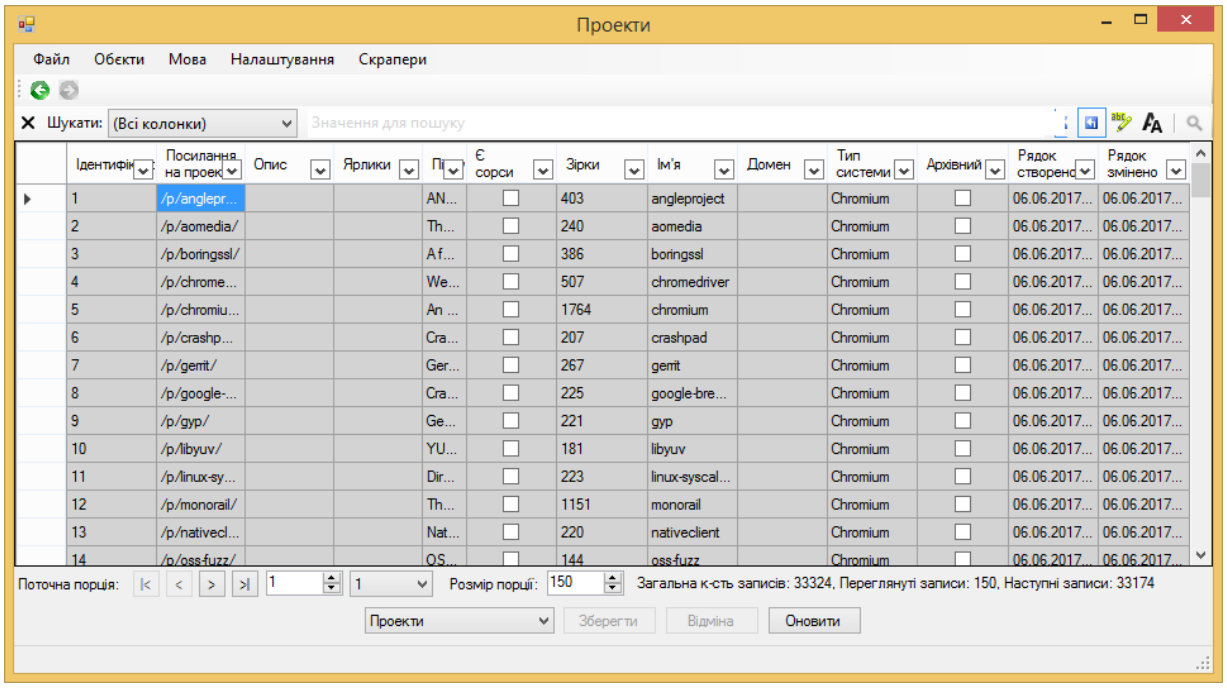


Рисунок 5 – Табличне представлення отриманих даних

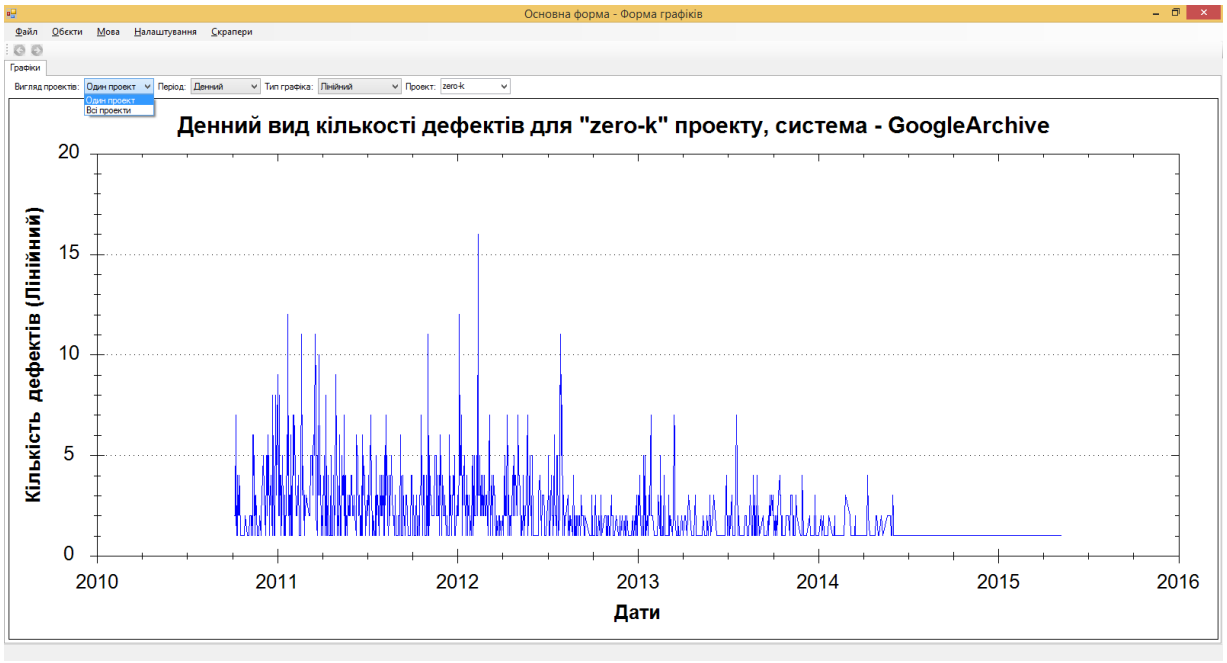


Рисунок 6 – Графічне відображення інформації про дефекти проекту в лінійному представленні

Користувач має можливість переключити графічне відображення даних з лінійного на акомулятивне. Дане представлення показано на рисунку 7. Тут користувач може побачити кількість дефектів за певний період часу, який також може бути змінений (денний, тижневий або місячний), що дає йому можливість відслідкувати стабілізацію кількості дефектів на проекті.

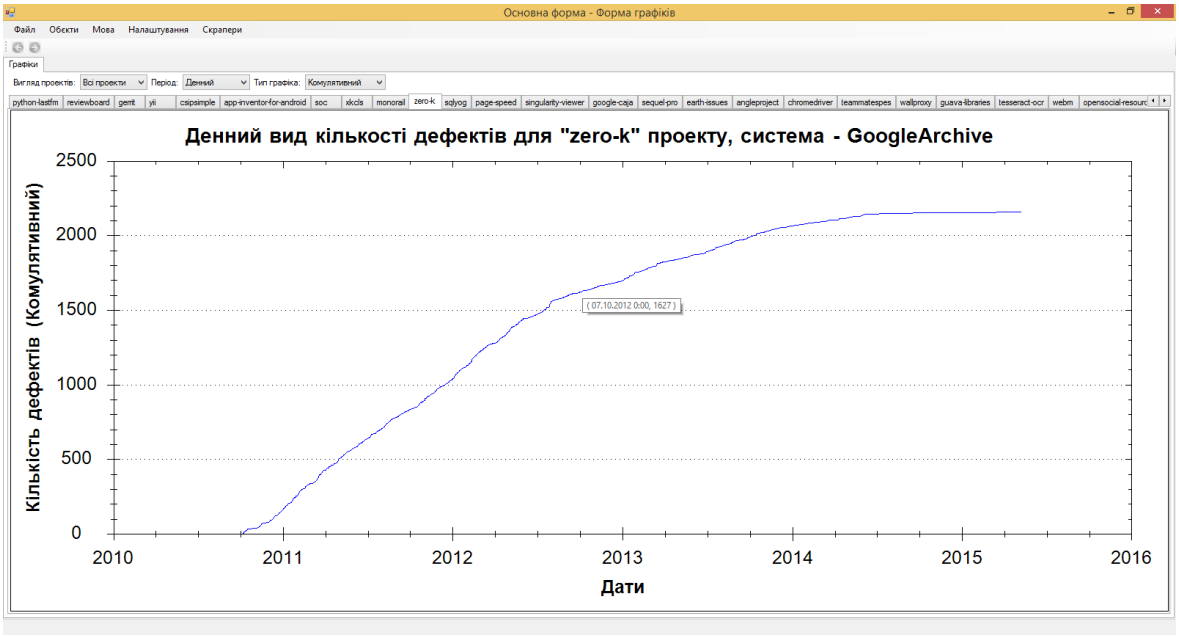


Рисунок 7 – Графічне відображення інформації про дефекти проекту в акомулятивному вигляді

Висновки

На сьогоднішній день моделі оцінювання надійності програмних систем здебільшого базуються (розроблені) на однотипних даних зібраних з одного джерела. У даній роботі описано потреба, вимоги і архітектура програмного комплексу, основною ціллю якого являється збір даних з різноманітних джерел, перетворення до єдиного формату і візуалізація даних про дефекти проектів для подальшого їх аналізу, перевірки і адаптації на відомих математичних моделях, визначення надійності системи на основі кумулятивної похибки.

Оскільки даний програмний комплекс спрямований на збір даних з різних джерел, це дасть можливість отримати потрібну, більш повноцінну інформацію для перевірки і адаптації існуючих методів моделювання надійності програмних систем на основі кумулятивної похибки.

Зібрана інформація буде використана в подальших дослідженнях для удосконалення відомих моделей надійності програмних систем за рахунок визначення додаткових коефіцієнтів, які дозволять використовувати вказані моделі на будь яких наборах вхідних даних.

Список літератури

[1] *ДСТУ 2844-94. Програмні засоби ЕОМ. Забезпечення якості. Терміни та визначення*. К. : Держстандарт України, 1996. – 19 с.

[2] H. Pham, System software reliability, Springer-Verlag: London Limited, 2006. – 440 p.

[3] S. Krepych, A. Dyvak, M. Dyvak, I. Spivak, "The method of providing of functional suitability of elements of the device of formation of signal in electrophysiological way of classification tissues surgical wound," on *13th International Conference Perspective Technologies and Methods in MEMS Design*, MEMSTECH 2017 Proceedings, pp. 183-186.

[4] С. Я. Крепич, «Метод синтезу смугового фільтра для заданих обмежень на його модуль коефіцієнта передачі,» на *Сучасні комп'ютерні інформаційні технології: Матеріали IV Всеукраїнської школи-семінару молодих вчених і студентів АСІТ'2014*. – Тернопіль: Економічна думка, 2014. – с. 26-29.

[5] S. Krepych, P. Stakhiv, and I. Spivak, «Analysis of the tolerance area parameters REC based on technological area scattering» on *12-th International Conference "The Experience Of Designing And Application Of CAD Systems in Microelectronics*". – 2013. – pp. 179–180.

[6] Р. В. Крепич «Особливості підходів до моделювання надійності програмних систем,» на VI Всеукраїнській школі-семінарі молодих вчених і студентів АСІТ'2016, 2016. с. 125–126.

[7] R. Peng, Y. F. Li, W. J. Zhang, and Q. P. Hu, Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction *Reliability Engineering and System Safety*. Vol. 126, p. 37–43, 2014.

[8] С. Я. Крепич «Моделювання та забезпечення функціональної придатності статичних систем методами аналізу інтервальних даних,» дис. канд. техн. наук, НУ «Львівська політехніка», Львів, 2016. 166 с.

[9] M.Dyvak, P. Stakhiv, I. Maksymova, and O. Potravych «Identification of the dynamic models by the adaptive method of tolerance estimation,» on *The Experience of Designing and Application of CAD Systems in Microelectronics - Proceedings of the 9th International Conference, CADSM 2007*, 2007. p. 365–368.

[10] М. П. Дивак, *Задачі математичного моделювання статистичних систем з інтервальними даними*, Тернопіль: Економічна думка, 2011. – 216 с.

[11] Б. В. Гнеденко, Ю. К. Беляев, и А. Д. Соловьев, *Математические методы в теории надежности*, М.: "Наука", 1965. – 524 с.

[12] В. С. Яковина «Методи та засоби аналізу надійності функціонування програмного забезпечення з урахуванням етапів життєвого циклу» : дис. докт. техн. наук, Львів, 2016. – 325 с.

[13] Monorail [Online] Available: https://bugs.chromium.org/

[14] Google Code [Online] Available: https://code.google.com/

[15] [Online] Available: https://storage.googleapis.com/google-code-archive/v2/

[16]System Dashboard [Online] Available: https://bugreports.qt.io/secure/Dashboard.jspa

[17] Э. Троелсен, *Язык программирования C# 6.0 и платформа .NET*, Диалектика-Вильямс. 2016. 1440 с.

Стаття надійшла: 28.05.2018.

Відомості про авторів

Крепич Роман Володимирович – аспірант кафедри комп'ютерних наук,
Крепич Світлана Ярославівна – к. т .н., старший викладач кафедри комп'ютерних наук.

R. V. Krepych[1], S. Y. Krepych[1]

COMPLEX SOFTWARE FOR COLLECTION AND VISUALIZATION OF SOFTWARE DEFECTS STATISTICS

[1]Ternopil National Economic University

УДК 004.8

Т. О. Савчук[1], Н. В. Приймак[1]

РОЗРОБКА ІНФОРМАЦІЙНОЇ МОДЕЛІ ПРОЦЕСУ ПОШУКУ АСОЦІАТИВНИХ ПРАВИЛ ПРИ РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

[1]Вінницький національний технічний університет

Анотація. У роботі сформовано задачу пошуку асоціативних правил при розробці програмного забезпечення, а також запропоновано інформаційну модель процесу пошуку асоціативних правил при розробці програмного забезпечення. При цьому, розглянуто моделі розробки програмного забезпечення, їх переваги та недоліки; вказано основні етапи даного процесу; обгрунтовано доцільність пошуку асоціативних правил при розробці програмного забезпечення, з метою знаходження залежностей, що можуть бути використані для визначення часу необхідного для виправлення завдання певним розробником. Менеджери проекту можуть використовувати дану інформацію для планування та управління процесом розробки програмного забезпечення. Доведено доцільність використання запропонованої інформаційної моделі процесу пошуку асоціативних правил при розробці програмного забезпечення. Результати проведеного дослідження показали, що інформативність асоціативних правил зросла, а тривалість їх пошук скоротилася. Розроблена інформаційна модель процесу пошуку асоціативних правил при розробці програмного забезпечення може бути використана при розробці відповідної інформаційної технології.

Ключові слова: програмне забезпечення; інформаційна модель; асоціативні правила; Data Mining.

Аннотация. В работе сформирована задача поиска ассоциативных правил при разработке программного обеспечения, а также предложена информационная модель процесса поиска ассоциативных правил при разработке программного обеспечения. При этом, рассмотрены модели разработки программного обеспечения, их преимущества и недостатки; указаны основные этапы данного процесса; обоснована целесообразность поиска ассоциативных правил при разработке программного обеспечения, с целью нахождения зависимостей, которые могут быть использованы для определения времени, необходимого для исправления задания определенным разработчиком. Менеджеры проекта могут использовать данную информацию для планирования и управления процессом разработки программного обеспечения. Доказана целесообразность использования предложенной информационной модели процесса поиска ассоциативных правил при разработке программного обеспечения. Результаты проведенного исследования показали, что информативность ассоциативных правил выросла, а продолжительность их поиск сократилась. Разработанная информационная модель процесса поиска ассоциативных правил при разработке программного может быть использована при разработке соответствующей информационной технологии.

Ключевые слова: программное обеспечение; информационная модель; ассоциативные правила; Data Mining.

Abstract. The problem of associative rules search during software development is formed in the work, and the informational model of the process of associative rules search during software development is proposed. In this paper, the models of software development, their advantages and disadvantages are considered; the main stages of this process are specified; the expediency of associative rules search during the software development was substantiated in order to find dependencies that can be used to determine the time required by a particular developer to correct a task. Project managers can use this information to plan and manage the software development process. The expediency of using the proposed information model of associative rules search during software development is proved. The results of the research showed that the informative of the associative rules has increased and the search time has decreased. The developed information model of associative rules search during software development can be used in the development of relevant information technology.

Keywords: software; information model; associative rules; Data Mining.
DOI: https://doi.org/10.31649/1999-9941-2018-42-2-43-48.

Вступ

Програмне забезпечення (ПЗ) – це набір комп'ютерних програм, представлених в цифровому вигляді, що використовуються для вирішення задач певного класу [1]. В свою чергу комп'ютерна програма – це упорядкована послідовність інструкцій, створених для досягнення поставлених цілей [1]. Процес розробки програмного забезпечення – це спланований та визначений процес створення комп'ютерних програм [2].

Актуальність

Процес розробки програмного забезпечення може бути здійснений з використанням різноманітних технологій розробки, в основу яких покладено такі моделі [2]:

1. Waterfall Model (каскадна модель) – модель, під час використання якої відбувається послідовна реалізація усіх стадій розробки, кожна з яких має повністю завершитися перед початком наступної. Дана модель зручна при керуванні невеликим проектом, тому що розробка ПЗ проходить швидко, вартість і терміни закінчення розробки заздалегідь визначені. Недоліком даної моделі є те, що вона може бути застосована лише при розробці ПЗ у якого технічне завдання не зазнаватиме змін у процесі розробки, що майже неможливо під час розробки програм.

2. V – подібна модель застосовується при розробці ПЗ для якого важливе безперебійне функціонування, наприклад програми координування безперервних технологічних процесів на виробництвах та атомних реакторах. Особливістю даної моделі є те, що тестування програми проводиться одночасно з відповідною стадією розробки: під час аналізу вимог, здійснюється перевірка на відсутність логічних помилок в діях користувачів, розглядаються всі можливі способи використання даного ПЗ, а під час кодування пишуться модульні тести. Недоліком є використання ресурсів на паралельне тестування.

3. Інкрементна модель передбачає кілька циклів розробки програмного забезпечення, що утворює життєвий цикл «мульти-каскад». Процедура розробки ПЗ згідно даної моделі передбачає випуск на першому етапі продукту в базовій функціональності, а потім послідовне додавання нових функцій, так званих «інкрементів». Серед недоліків даної моделі можна виділити: необхідність планування та дизайну ПЗ, що розробляється, чіткого і повного визначення усієї програми.

4. Ітеративна модель передбачає створення частини функціоналу, що стає базою для визначення подальших вимог. Для успішного використання ітеративної моделі розробки програмного забезпечення потрібно здійснювати перевірку вимог до кожної версії програмного забезпечення в рамках кожного циклу моделі. Недоліками даної моделі є необхідність активного управління процесом розробки ПЗ, неможливість визначення точної дати завершення розробки, необхідність передбачення та аналізу можливих ризиків.

5. Agile model (гнучка модель розробки) характеризується тим, що після кожної ітерації процесу розробки ПЗ, замовник може спостерігати результат і розуміти, задовольняє він його чи ні. Серед недоліків даної моделі виділяють відсутність сформульованого очікуваного результату та складність оцінки трудовитрат і вартості розробки.

Порівняльний аналіз моделей процесу розробки програмного забезпечення наведено у табл. 1.

Таблиця 1 – Моделі процесу розробки програмного забезпечення

Назва моделі	Переваги	Недоліки
Каскадна модель	Зручна для створення невеликого проекту	Можна застосовувати лише при розробці ПЗ із технічним завданням, яке не змінюватиметься у процесі розробки
V – подібна модель	Кожен етап розробки ПЗ тестується відповідними засобами	Необхідно використовувати ресурси на паралельне тестування
Інкрементна модель	Можна побачити завершений базовий функціонал програмного продукту після першої ітерації розробки	Необхідно здійснювати планування та дизайн ПЗ, чітко і повно визначити програму, перш ніж вона буде розроблена
Ітеративна модель	Під час кожної ітерації розробки ПЗ, здійснюється перевірка його вимог та версій	Необхідно активно управляти процесом розробки ПЗ; неможливо визначити точну дату завершення розробки; необхідно провести аналізі можливих ризиків
Agile model	Після кожної ітерації розробки, замовник може спостерігати результат і розуміти, задовольняє він його чи ні	Відсутність сформульованого очікуваного результату; складність в оцінці трудовитрат і вартості розробки

Під час розробки ПЗ накопичується велика кількість інформації, результати аналізу якої, можна використовувати для управління та удосконалення процесу розробки ПЗ. Для аналізу збереженої інформації доцільно застосовувати технології Data mining, що розроблені з метою отримання корисних (з точки зору експерта) даних, залежностей, шаблонів тощо зі збереженої інформації.

Тому, актуальною є задача розробки інформаційної моделі пошуку асоціативних правил під час розробки програмного забезпечення. Знайдені асоціативні залежності можна використовувати при плануванні, передбаченні та розумінні процесу розробки програми, а також для підтримки та управління даного процесу.

Мета

Метою даного дослідження є розробка інформаційної моделі процесу пошуку асоціативних правил при розробці програмного забезпечення, використання якої дозволить скоротити час пошуку асоціативних правил (АП) та підвищити їх інформативність.

Задачі

1. Сформувати задачу пошуку асоціативних правил при розробці програмного забезпечення
2. Розробити інформаційну модель процесу пошуку асоціативних правил під час розробки програмного забезпечення

Розв'язання задач

Нехай Dev_j – це j-ий розробник, $Task_i$ – це i-те завдання, що має ряд характеристик та буде ви-

рішене j-м розробником, а t_{i_j} – час, необхідний на розв'язання i-го завдання j-им розробником.

Задачу пошуку асоціативних правил при розробці програмного забезпечення можна описати наступним чином:

$$Task_i \cup Dev_j \rightarrow t_{i_j},\ Task_i \cup Dev_j \cap t_{i_j} \rightarrow \varnothing$$

Тобто, необхідно знайти асоціативні правила виду: «якщо завдання $Task_i$ буде розв'язуватися розробником Dev_j, то йому потрібно буде t_{i_j} часу».

Визначимо інформаційну модель процесу пошуку асоціативних правил при розробці програмного забезпечення, що відображає вхідні та вихідні значення даного процесу, важливі параметри і величини, а також їхні зв'язки.

Відповідну інформаційну модель можна подати у вигляді кортежа $IMARM$:

$$IMARM = \langle Task, Dev, minsupp, minconf, method, AR, newTask, PredictTask \rangle, \quad (1)$$

де $Task$ – множина завдань, що були виконані під час розробки ПЗ; Dev – множина розробників, які можуть виконати поставлене завдання; $minsupp$ – мінімальне значення підтримки АП; $minconf$ – мінімальне значення достовірності АП; $method$ – метод, за допомогою якого буде здійснюватися пошук АП; AR – множина знайдених асоціативних правил; $newTask$ – множина завдань, для яких необхідно визначити тривалість їх реалізації певним розробником; $Predict\ Task$ – множина завдань з часом, необхідним на їх виконання певним розробником.

Нехай i-те завдання $Task_i$ описується:

$$Task_i = \langle Type, Priority, Severity, Component, t_{i_j}, Dev_j \rangle, \quad (2)$$

де $i = \overline{1,n}$, n – кількість завдань; $Type$ – це тип i-го завдання, що має наступну множину значень {покращення, особливість, дефект}. В залежності від типу завдання, воно виконується на конкретному етапі розробки ПЗ. $Priority$ – це пріоритет i-го завдання, що має наступну множину значень {високий, середній, низький} та описує важливість і порядок, в якому потрібно виконати завдання у порівнянні з іншими. $Severity$ – це важливість i-го завдання, що має наступну множину значень {блокуюча, критична, важлива, помірна, незначна}. Дана характеристика визначає рівень впливу даного завдання на функціонування ПЗ в цілому. Даний показник важливий для менеджерів проектів, оскільки він є основним при вирішенні бізнес-питань. $Component$ – це компонент розроблюваного ПЗ, множина значень якого залежить від конкретного ПЗ, тобто це частина програмного забезпечення, для якої буде виконуватися завдання певного типу. t_{i_j} – це час, необхідний для виконання i-го завдання j-м розробником. Значення даної величини залежить від конкретного завдання та розробника, що буде його виконувати. Менеджери проекту повинні знати величну даної характеристики для планування процесу розробки ПЗ та ресурсів, необхідних для нього.

Знайдені асоціативні правила при розробці ПЗ утворюють множину AR. Нехай d-те асоціативне правило описується:

$$AR_d = \{ Task_i \cup Dev_j \rightarrow t_{i_j},\ Task_i \cup Dev_j \cap t_{i_j} \rightarrow \varnothing \}, \quad (3)$$

де $d = \overline{1,c}$, d – кількість знайдених асоціативних правил.

Множину $newTask$, утворюють завдання, які потрібно виконати, під час розробки ПЗ і для яких не визначено тривалість їх виконання. Нехай f-те завдання $newTask_f$ описується:

$$newTask_f = \langle Type, Priority, Severity, Component \rangle, \quad (4)$$

де $f = \overline{1,g}$, g – кількість завдань.

Множина $Predict\ Task$, складається із завдань, для яких визначено тривалість їх виконання. Нехай l-те завдань з часом, необхідним на його виконання певним розробником $Predict\ Task_l$ описується:

$$PredictTask_l = \left\langle Type, Priority, Severity, Component, Dev_j, t_{l_j} \right\rangle. \quad (5)$$

де $l = \overline{1,k}$, k – кількість завдань; t_{l_j} – це тривалість виконання l-го завдання j-им розробником.

Запропонована інформаційна модель процесу пошуку асоціативних правил при розробці програмного забезпечення розділена на два окремих блоки (рис. 1):

1. Блок пошуку асоціативних правил – де описано процес пошуку асоціативних правил.

Результатом виконання даного процесу є множина асоціативних правил AR, що використовуються для визначення часу, необхідного на виконання завдання конкретним розробником у наступному блоці.

Для пошуку асоціативних правил при розробці ПЗ використовується удосконалений алгоритм Frequent Pattern Growth (FPG) [3, 4], що був модифікований за рахунок класифікації завдань на три групи в залежності від складності їх виконання. Така модифікація дозволяє пришвидшити пошук АП, оскільки він здійснюватиметься паралельно в створених групах, та підвищити їх інформативність.

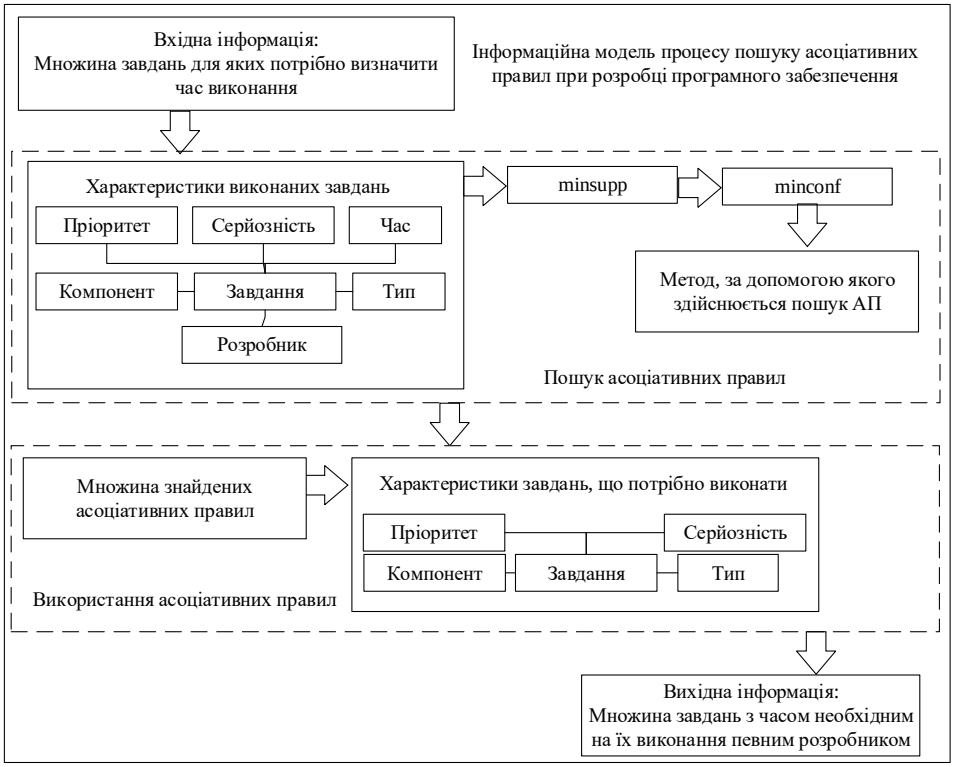


Рисунок 1 – Інформаційна модель процесу пошуку АП під час розробки програмного забезпечення

Значення мінімальної підтримки $minsupp$ і мінімальної достовірності $minconf$ є обов'язковими при пошуку асоціативних правил і задається експертом [5]. Неправильно визначена величина даних параметрів, може призвести до невірної генерації АП: при встановленій високій мінімальній підтримці може бути згенеровано лише кілька правил, але багато часу буде витрачено на сканування бази даних. Вибір низької мінімальної підтримки може призвести до численних надлишкових правил. Тому виникає необхідність у виборі оптимального значення для даних показників.

Значення підтримки асоціативного правила $X \to Y$ дорівнює значенню кількості транзакцій в яких X і Y зустрічаються разом [6].

Для визначення значення мінімальної підтримки $minsupp$ для пошуку АП під час розробки ПЗ за-

стосуємо функцію, відображену у виразі 1.6 [7]. Ідея використання цієї функції для пошуку частих предметних наборів була вперше запропонована в [8], що дозволяє встановити високе значення мінімальної підтримки, коли даних в БД небагато, а потім зменшувати його, у випадку збільшення кількості даних:

$$minsupp = (e^{(-\ ax-\ b)}) + c, \tag{6}$$

де x – це кількість записів в БД; a, b, c – додатні константи. Константа c – це найменше значення, якому може дорівнювати достовірності асоціативного правила. Константи a та b впливають на те, як різко повинна зменшуватися $minsupp(x)$, коли x збільшиться.

2. У наступному блоці інформаційної моделі процесу пошуку АП під час розробки ПЗ відображено процес використання асоціативних правил, з метою визначення часу, необхідного на виконання завдання конкретним розробником.

Результатом даного процесу є множина завдань $Predict\ Task_l$ з часом, необхідним на їх реалізацію певним розробником.

Оскільки процес розробки ПЗ розглядається з точки зору менеджера проекту, то необхідно вирішити задачу прийняття рішень щодо відбору завдань, які можна реалізувати у визначені терміни та з використанням вказаного бюджету.

Такий план можна представити кортежом $PLAN$ вираз 1.7:

$$PLAN = \langle PredictTask, Budget, Duration, Fault \rangle, \tag{7}$$

де $Budget$ – бюджет, виділений на розробку ПЗ. Це максимальна сума коштів, що може бути витрачена на розробку даного програмного продукту чи його складової. Величина даної характеристики відрізняється в залежності від проекту. $Duration$ – запланована тривалість розробки ПЗ. Величина даної характеристики також залежить від проекту. $Fault$ – допустиме відхилення тривалості та вартості розробки, що встановлюється менеджером проекту, в залежності від проекту.

Ефективність використання розробленої інформаційної моделі процесу пошуку асоціативних правил при розробці програмного забезпечення було підтверджено результатами проведеного експерименту.

Експеримент проводився з наступною тестовою інформацією:

- кількість завдань – до 100 елементів, до 500 елементів, до 1000 елементів, до 1500 елементів;
- кількість знайдених асоціативних правил;
- час, витрачений на їх пошук в секундах.

Результати експерименту з використанням запропонованої інформаційної моделі та без неї представлені на рис. 2:

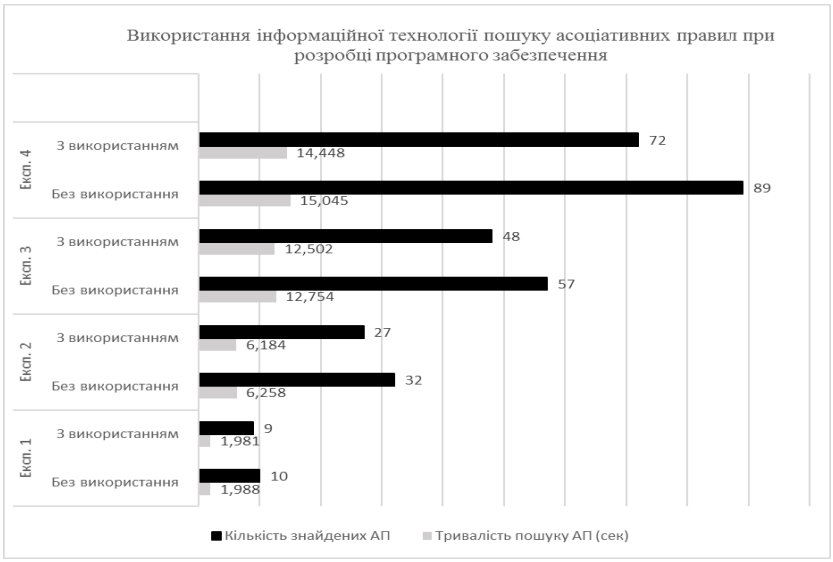


Рисунок 2 – Результати використання інформаційної моделі процесу пошуку АП під час розробки ПЗ

1. Зі збільшенням кількості завдань, серед яких здійснювався пошук АП, збільшується час, необхідний для їх пошуку; але різниця між часом, необхідним на пошук асоціативних правил з використанням розробленої інформаційної моделі та без неї, збільшується зі збільшенням загальної кількості тестових даних.

2. Швидкість пошуку АП з використанням запропонованої інформаційної моделі зросла наступним чином: кількість елементів до 100 – зросла на 0,35%, кількість елементів до 500 – на 1,2%, кількість елементів до 1000 – на 2%, кількість елементів до 1500 – на 3,96%.

3. Інформативність та цінність знайдених асоціативних правил при використанні запропонованої інформаційної моделі вища, оскільки кількість таких АП менша ніж без використання відповідної інформаційної моделі.

Висновки

1. Сформовано задачу пошуку асоціативних правил під час розробки програмного забезпечення,

2. Розроблено інформаційну модель даного процесу, що може бути використана при розробці відповідної інформаційної технології. Використання запропонованої інформаційної моделі дозволило скоротити час пошуку асоціативних правил та підвищити їх інформативність, що підтверджено отриманими результатами проведених експериментів.

Список літератури

[1] В. К. Батоврин, Толковый словарь по системной и программной инженерии: М: ДМК Пресс, 2012.

[2] Application Development, 2014. [Online]. Available: http://www.bestpricecomputers.co.uk/glossary/application-development.htm.

[3] Т. О. Савчук, Н. В. Приймак, «Використання fpg-алгоритму для пошуку асоціативних правил при прийнятті рішень в управлінні процесами», на XLVI НТК професорсько-викладацького складу, співробітників та студентів університету, Вінниця, 2017.

[4] Т. О. Савчук, Н. В. Приймак, «Обгрунтування вибору методу генерації частих предметних наборів для пошуку асоціативних правил при розробці прогрмного забезпечення», на XI міжнародній конференції «Інтернет-освіта-наука-2018», Вінниця, 2018, с.45-46.

[5] Ассоциативные правила, 2016 [Електронний ресурс]. Режим доступу: https://www.researchgate.net/publication/315629798_AssociativnyepravilSravnitelnyjnalinstrumentaria.

[6] J. Manimaran, "Analysing the quality of Association Rules by Computing an Interestingness Measures", Indian Journal of Science and Technology, Vol 8(15), pp. 1-12, July. 2015. DOI: 10.17485/ijst/2015/v8i15/76693

[7] P. Fournier-Viger, How to auto-adjust the minimum support threshold according to the data size, 2014. [Online]. Available: http://data-mining.philippe-fournier-viger.com/how-to-auto-adjust-the-minimum-support-threshold-according-to-the-data-size/.

[8] P. Fournier-Viger, "Un modèle hybride pour le support à l'apprentissage dans les domaines procéduraux et mal-définis". Ph.D. Thesis, University of Quebec in Montreal, Montreal, 2010.

Стаття надійшла: 28.08.18.

Відомості про авторів

Савчук Тамара Олександрівна – PhD, професор кафедри комп'ютерних наук Вінницького національного технічного університету.

Приймак Наталія Василівна – аспірант кафедри комп'ютерних наук Вінницького національного технічного університету.

T. O. Savchuk[1], N. V. Pryimak[1]

DEVELOPMENT OF THE INFORMATION MODEL OF THE ASSOCIATIVE RULES SEARCH DURING THE SOFTWARE DEVELOPMENT

[1]Vinnytsia National Technical University

УДК 378.21

А. А. Яровий[1], С. В. Барабан[1], Р. В. Криночкін[1]

ІНТЕЛЕКТУАЛЬНИЙ МОДУЛЬ БРАУЗЕРНОЇ СИСТЕМИ УПРАВЛІННЯ ІТ-ПРОЕКТАМИ TRELLO

[1]Вінницький національний технічний університет

Анотація. В даній статті запропоновано програмне рішення у вигляді інтелектуального модуля для розширення функціоналу браузерної системи управління ІТ-проектами Trello на основі використання її API. Програмування здійснено мовою python з використанням поширених бібліотек. Для відбору даних з дошок, списків, карток Trello були розроблені функції за допомогою використання методів процедурного програмування. Під час розробки інтелектуального модуля проведено класову декомпозицію програмного коду, розроблено UML-діаграму класів інтелектуального модуля відповідно до існуючої бібліотеки py-trello. Для агрегації даних, що були відібрані з дошок, списків, карток Trello використано сучасну python бібліотеку Pandas. При цьому застосовувалися такі функції Pandas як зчитування, запис до файлу у форматі Microsoft Excel, групування даних, перезавантаження індексу таблиці, арифметичні додавання і множення. Апробація результатів роботи розробленого інтелектуального модуля системи управління Trello підтвердила доцільність розробки при здійснені щомісячної звітності та обчислення метрик продуктивності роботи працівників.

Ключові слова: Trello, py-trello, python, pandas, API, система управління, функція, ООП, дошка, список, картка, агрегація даних.

Аннотация. В данной статье предлагается программное решение в виде интеллектуального модуля для расширения функционала браузерной системы управления ИТ-проектами Trello на основе использования ее API. Программирование проведено на языке python с использованием распространенных библиотек. Для отбора данных из досок, списков, карточек Trello были разработаны функции посредством использования методов процедурного программирования. При разработке интеллектуального модуля проведена классовая декомпозиция программного кода, разработана UML-диаграмма классов интеллектуального модуля в соответствии с существующей библиотекой py-trello. Для агрегации данных, которые были отобраны из досок, списков, карточек Trello использовано современную python библиотеку Pandas. При этом применялись такие функции Pandas как считывания, запись в файл в формате Microsoft Excel, группировка данных, перезагрузка индекса таблицы, арифметические сложения и умножения. Апробация результатов работы разработанного интеллектуального модуля системы управления Trello подтвердила целесообразность разработки при осуществлении ежемесячной отчетности и вычисления метрик производительности работы сотрудников.

Ключевые слова. Trello, py-trello, python, pandas, API, система управления, функция, ООП, доска, список, карточка, агрегация данных.

Abstract. In the article was proposed a programmatic solution in the form of an intelligent module for expanding the functionality of a browser-based management system for IT-projectsTrello based on the use of its API. To select data from the boards, lists, cards, functions were developed using procedural programming methods. During the development of the intellectual module, a UML-class diagram of the intellectual module developed according to the existing py-trello library. For aggregation of data, selected from the boards, lists, cards, the modern Pandas library is used. The Pandas features such as reading, writing to a Microsoft Excel file, data grouping, reloading the table index, arithmetic adding and multiplication were used. Approval of the results confirmed the feasibility of the development of the implementation of monthly reporting and calculation of employee productivity metrics.

Key words: Trello, py-trello, python, pandas, API, control system, function, OOP, board, list, card, data aggregation.
DOI: https://doi.org/10.31649/1999-9941-2018-42-2-49-54.

Вступ

Система управління проектами є невід'ємною складовою сучасної розробки програмних продуктів і використовується не тільки в ІТ-компаніях різної величини, а й більшістю фрілансерів [1]. Таке розповсюдження стало можливим завдяки наданню проектно-орієнтованого робочого простору для ведення одного чи більше проектів, при цьому доступ є у всіх учасників проекту. Систем управління проектами є надзвичайно багато, з різними можливостями, характеристиками і для будь-якого користувача з будь-яким бюджетом, адже навіть платні системи надають хоча й обмежені, але freeversionвласного продукту. Все одно найбільш перспективним є web-basedсистеми, не дивлячись на те, що компільовані варіанти в деяких випадках більш продуктивні, однак менш гнучкі для кастомізації.

Найбільш поширеною в даний час системою управління проектами є Jiгакомпанії Atlassian, хоча вона розроблялася з самого початку як багтрекер – система відстеження помилок, але поступово доповнилася функціями організації спілкування з користувачами та управлінням проектами. Таким чином, вона підходить для керування проектами та помилками в коді, але вона платна. Існує безкоштовна версія, проте значно обмежена у функціоналі. Jira підходить для великих ІТ-компаній, а для малих компаній або просто команди програмістів не є оптимальним варіантом. Тому попрацювавши з Jira, ІТ-компанія SinaiR&D вирішила перейти на щось інше, а саме Trello[2].Trello – безкоштовна браузерна система управління проектами, що використовує парадигму керування проектами, відому як канбан. Проте, щоб повністю замінити Jira, необхідно доробити обгортку навколо Trello, чому і присвячена дана стаття.

Мета

Метою роботи є розширення функціоналу браузерної системи управління ІТ-проектами Trello за рахунок розробки інтелектуального модуля до API Trello.

Задачі

1) відбір даних з дошок, списків і карток Trello;
2) агрегація отриманих даних;
3) виведення даних у вигляді таблиць у файлі формату MicrosoftExcel;
4) розробка інтелектуального модуля до API Trello;
5) апробація результатів розробки в діяльності IT-команди програмістів SinaiRnD.

Розв'язання задач

Trello є прикладом реалізації простої системи канбан. Канбан було розроблено ученим Таїті Оно, в Toyota, з метою досягнення та підтримки високого рівня виробництва. Канбан став ефективним інструментом в управлінні системою виробництва у цілому, і довів себе, як відмінний спосіб пропагування вдосконалень. Картки є ключовим компонентом канбану і сигналізують про необхідність переміщення задач усередині проекту. Згідно цієї системи у Trelloє дошки, списки і картки. Інтелектуальний модуль повинен відбирати назви дошок, назви груп, в які входять дошки, назви карток, ідентифікаційні номера карток, повні імена членів карток, задані користувачем поля в картках [3]. Назви дошок в даному випадку – це назви проектів, групи, в які входять дошки – це назви організацій проектів, назви карток – це є задачі, виконання яких призводить до прогресу проекту, повні імена членів карток – це імена розробників, що виконують поставлені задачі. Найбільш інформативним видом даних є користувацькі поля в картках. Для обліку об'єму зусиль, що знадобились для виконання задачі певного проекту використовується поле – "SP"("Story Points"), яке являє собою список з чисел Фібоначчі від 1 до 100, а також поле "Payed", що використовується для полегшення фільтрації задач, які вже оброблені системою. Поле "SP" необхідне для нарахування балів розробникам за виконані задачі, на основі чого після агрегації даних має нараховуватися зарплатня. "Payed" необхідне для перевірки чи було оплачене конкретне завдання розробнику.

Бібліотека py-trello[4], яку ми вибрали за основу для нашої розробки побудована на основі класів. Об'єктами системи Trelloє board, list, card, organization, member, які є екземплярами класів Board, List, Card, Organization, Member бібліотеки py-trello. Для процедурного програмування [5] ми створимо змінні board(дошка), list(список), card(карточка), organization(група дошок), member(розробник), а також розробимо функції, які будуть використовувати ці змінні та методи класів бібліотеки py-trello.Найперша функція програмного модуля має отримувати дошку користувача, а повертати усі карточки цієї дошки, що знаходяться у списку виконаних ("Done"). Схему алгоритму роботи даної функції зображено на рис. 1. Функції list_lists() є полем класу Boardбібліотеки py-trello, а функція list_cards() є полем класу Card (рис. 1). Наступна функція перевіряє чи співпадає ідентифікаційний номер користувача карточки з ідентифікаційним номером користувача дошки і якщо присутня така рівність, то повертає змінну з даними користувача. Використовує ця функція дошку та ідентифікаційний номер, в якості вхідних даних, а також метод класу Boardget_members(). У випадку не співпадіння ідентифікаційних номерів користувача картки і дошки, функція повертає None. Дану функцію можна розширити або написати іншу, що повертатиме повне ім'я користувача картки.

Далі потрібно провести роботу з полями користувача в картках. У класу Card є метод get_custom_field_by_name(), що повертає дані customfield картки за іменем. Було створено дві функції, що по реалізації майже ідентичні – get_sp()і get_payed().get_sp() – приймає картку, використовує функцію get_custom_field_by_name() і повертає змінну, в якій записано значення поля "SP"в картці. Якщо в полі "SP" якимось чином опиниться значення None, функція поверне цифру 0. Функція get_payed() на вході приймає картку, використовує функцію get_custom_field_by_name() і повертає TRUE, якщо в мітці "Payed" картки стоїть галочка або повертає пустий рядок в іншому випадку.

Для агрегації даних використовуються популярні бібліотеки для роботи з масивами даних NumPy та Pandas [6]. Перед початком роботи необхідно імпортувати ці бібліотеки до програми. Вони були імпортовані як np та pd відповідно. В основному тілі програми створено три DataFrame – один для даних, відібраних з дошок, списків, карток Trello, другий – для агрегованих даних, а третій DataFrame – проміжний, для зберігання даних з excel-файлу, що містить вагові коефіцієнти працезатрат за проекти. Кінцевий варіант першого DataFrame має вигляд:

Таблиця 1 – DataFrame, що утворюється внаслідок виконання програми

	Assignee	Key	Points	Project	Summary
0	Roman	54321abcd	5	demo	Check if PyTrello supports Custom Fields
1	Serhii	145abd	13	demo	Getting known of Python API
2	Serhii	125acd	5	demo	Userstories
3	Serhii	15ad	55	demo	Printtablewith DATA

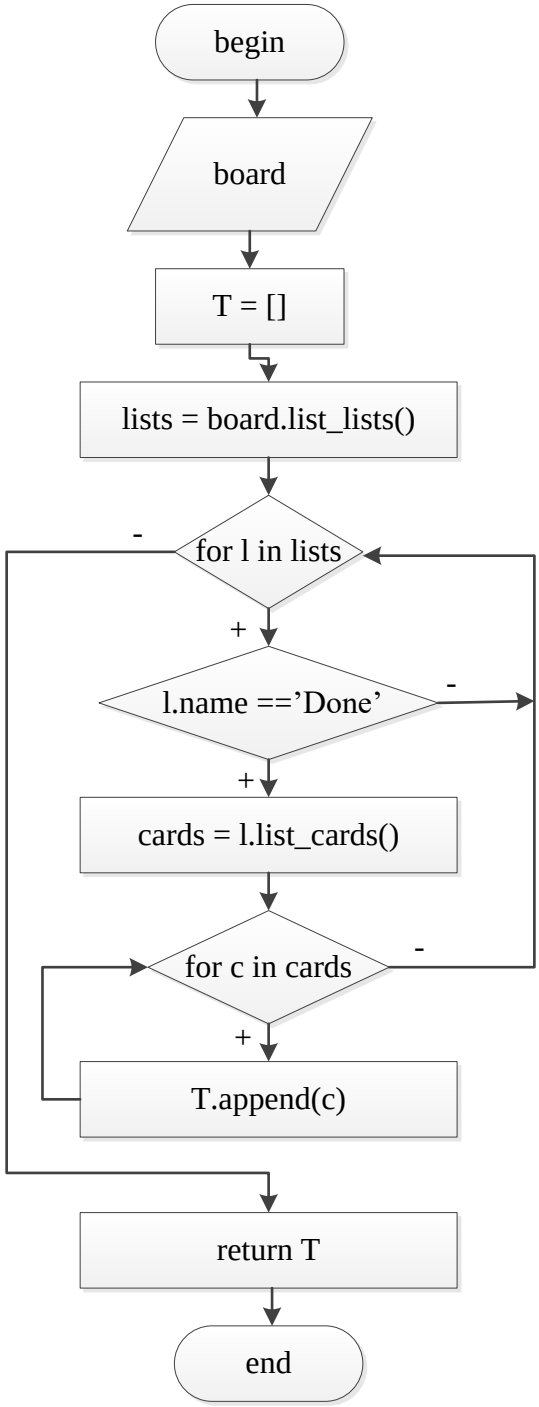


Рисунок 1 – Схема алгоритму роботи функції відбору карток з дошки

Для агрегації даних були використані наступні стандартні функції з бібліотеки Pandas: групування – grouby(), знаходження арифметичної суми даних – sum(), повернення індексів таблиці до попереднього стану – reset_index(), об’єднання декількох DataFrame – merge(). Зручністю програми є те, що всі функції можна використати в одному рядку, т.зв. “one-liner”:

```
aggregation = array[['Project', 'Assignee', 'Points']].groupby(['Assignee', 'Project']).sum().reset_index()
```

Зчитування даних з excel-файлу в Pandas бібліотеці також досить просте:

```
table = pd.read_excel('ProjectUCP.xlsx', 'Лист1')
```

Кінцевий варіант таблиці з агрегованими даним буде має вигляд:

Таблиця 2 – DataFrame з агрегованими даними

	Assignee	Project	Points	Coeff	Value
0	Serhii demo	15	5,5	82,5	
1	Roman demo	5	5,5	27,5	

Програмний код запис до excel-файлу в декілька листів буде мати вигляд:

```
writer = pd.ExcelWriter('third.xlsx')
array.to_excel(writer, "June")
aggregation.to_excel(writer, "June_agregation")
writer.save()
```

Проектування інтелектуального модуля за методологією ООП

Бібліотека py-trello, яку ми використовуємо за основу в розробці нашого програмного модуля і яка є обгорткою навколо API Trello, написаною на мові програмування Python, є об'єктно-орієнтованою. Тому розробка програмного модуля на основі класів є найбільш логічним вирішенням поставленої задачі.

Першим варіантом є створення класу DataFromTrello(), який би містив в якості методів усі ті функції, що були розроблені вище. Частина коду для розробленого класу DataFromTrello() буде мати вигляд:

```
class DataFromTrello():
def __init__(self):
self.client = TrelloClient(api_key="...", api_secret="...", token="...")
self.boards = self.client.list_boards(board_filter='starred')
self.array = pd.DataFrame(np.object, index=[], columns=[])
```

Далі код класу DataFromTrello() буде містити всі ті ж функції, розроблені для py-trello за методологією функціонального програмування. Створення екземпляру класу(об'єкту) в тілі програми:

```
a = DataFromTrello()
```

Використання екземпляру класу DataFromTrello() в тілі програми:

```
for b in a.boards:
   c = a.cards_from_board(b)
for card in c:
      s = a.sp_custom_field(card)
      m = a.name_member(b, card)
payed = a.payed_custom_field(card)
```

У цієї програми буде одна особливість. Клас DataFromTrello() не містить ніяких полів. Це означає, що він є статичним класом. Використання статичних класів для вирішення подібних до нашої задач не є типовим для мови програмування Python[7], а також для методології ООП розробки програмного забезпечення. Тому це не є оптимальним вирішенням задачі розробки програмного модуля системи управління проектами Trello.

Можливим варіантом розробки програмного модуля є використання наслідування класів Board і Card. Тоді постає необхідність створення обгортки для класу TrelloClient.

Більш оптимальним варіантом є використання в тілі програми об'єктів board і card, які будуть екземплярами класів BoardTrello і CardTrello відповідно. На початку розробки програмного модуля було створено UML-діаграму для відображення усіх класів з їх полями і методами для вирішення поставленої у меті задачі. На рис. 2 зображено дану діаграму.

З рис. 2 видно, що клас TrelloClient знаходиться ніби окремо від усіх інших і не пов'язаний з ними, але це клас, що містить поля і методи аутентифікації клієнта системи управління Trello. Напряму він не пов'язаний з усіма класами, але без нього неможлива робота ні одного класу. Класи Board, Card і Member наслідують клас TrelloBase. Таким чином будь-яка зміна класу TrelloBase призведе до необхідності зміни в усіх класів, що його наслідують. Відношення класу BoardTrello до класу Board так як і класу CardTrello до класу Cardє композиція, оскільки BoardTrello є частиною цілого Board, а CardTrello є лише частиною цілого Card. При цьому клас BoardTrello залежить від класів Cardi Member, а клас CardTrello залежить від класу Member. Відношення залежність зображено на рис. 2 пунктирною лінією. В основному тілі програми утворюються два об'єкти – об'єкт board, що є екземпляром класу BoardTrello та об'єкт card, що є

екземпляром класу CardTrello. Між даними двома об'єктами в тілі програми встановлюється тісний зв'язок. Саме тому відношення між класами BoardTrello і CardTrelloє асоціацією. Точніше - відношення між класами BoardTrello і CardTrello є N-арною асоціацією, оскільки одному об'єкту board відповідає множина об'єктів card. Тому на рис. 2 класи BoardTrello і CardTrello з'єднані суцільною лінією, а потужність зв'язку у них 1..*.

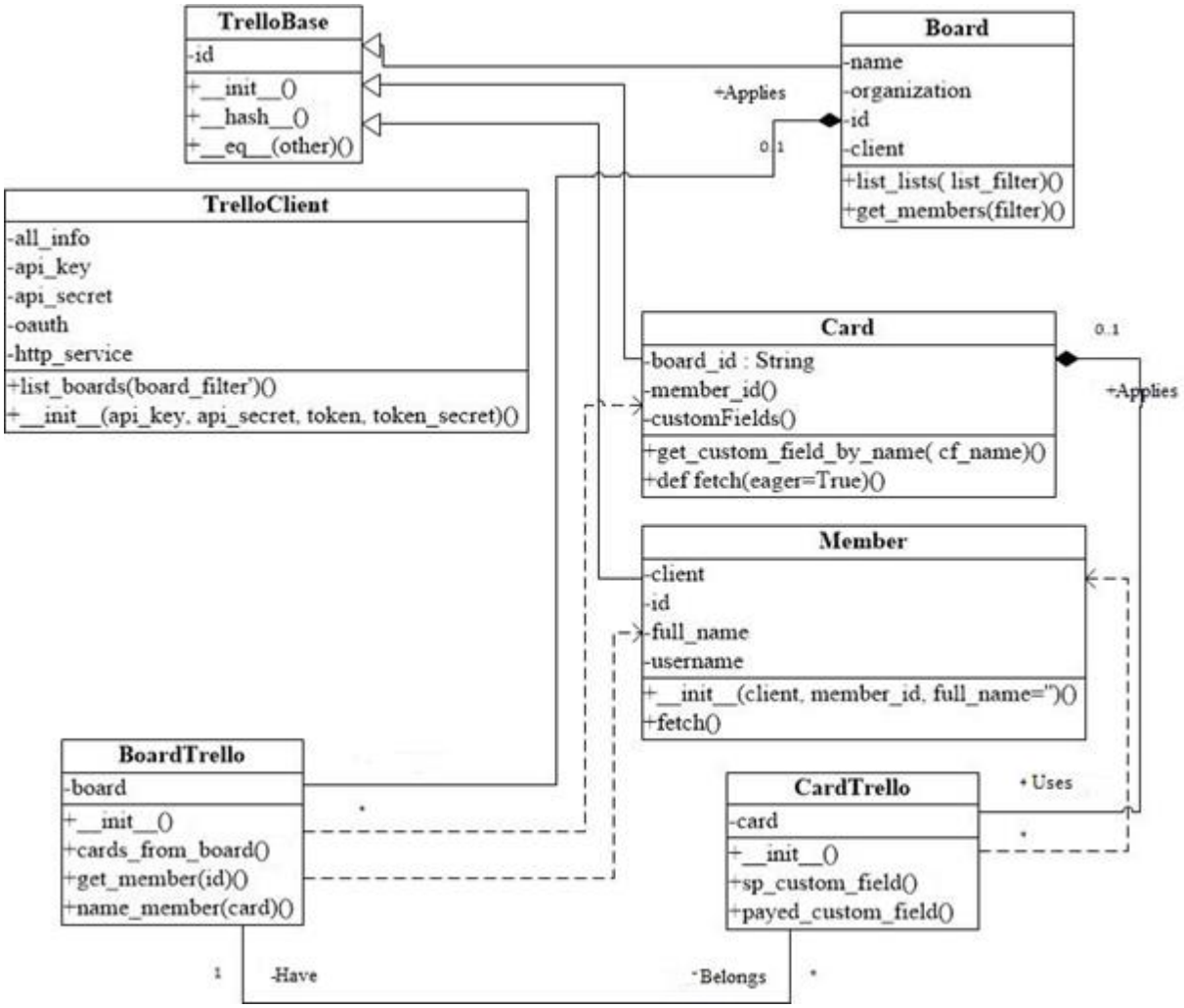


Рисунок 2 – UML-діаграма класів розробленого програмного модуля

Передача board у клас BoardTrello:

```
classBoardTrello():
def __init__(self, board):
    self.board = board
```

Обов'язково при передачі board і card необхідно включити перевірку чи ми отримали дійсно board з класу Board і card з класу Card, а якщо ні, то що за тип даних в отриманих board і card. На мові програмування Python це досить легко робиться:

```
assertisinstance(board, Board), "Wrong board type, while expected Board: {}".format(type(board))
```

це для board, а для card:

```
assertisinstance(card, Card), "Wrong card type, while expected Card: {}".format(type(card))
```

Тоді виклик екземплярів класів BoardTrello і CardTrello можна здійснити безпосередньо в тілі програми в місці, де вони будуть використовуватися:

```
for b in boards:
```

```
   a = BoardTrello(b)
   c = a.cards_from_board()
for card in c:
      d = CardTrello(card)
      s = d.sp_custom_field()
      m = a.name_member(card)
payed = d.payed_custom_field()
```

Після чого звісно ж йде заповнення DataFrame і запис, отриманих даних до Microsoft Excel, завдяки можливостям Python бібліотеки Pandas.

Висновки

1. Запропоновано програмне рішення розширення функціоналу браузерної системи управління проектами Trello на основі використання її API. Програмування проведено мовою Python. Апробація результатів підтвердила доцільність розробки при проведені щомісячної звітності та обчислення метрик продуктивності роботи працівників.

2. Поставлені в технічному завдані задачі були розв'язані методами процедурного програмування, на основі чого реалізовано функції, що лягли в основу роботи розробленого інтелектуального модуля браузерної системи управління проектами Trello.

3. Розроблено UML-діаграму класів інтелектуального модуля, на основі якої проведено класову декомпозицію розробленого програмного модуля і впорядковано код програми відповідно до існуючої бібліотеки py-trello.

4. Виконано агрегацію даних, відібраних з дошок, списків, карток Trello,з використанням сучасної python бібліотеки Pandas. При цьому було застосовано такі функції Pandas як зчитування, запис до файлу у форматі Microsoft Excel, групування даних, перезавантаження індексу таблиці, арифметичні додавання і множення.

Список літератури

[1] С. В. Поперешняк, «Проблеми підготовки IT-спеціалістів», *Системи обробки інформації*. № 7 (88), с. 127-131, 2010.

[2] David M. Beazley, Python Essential Reference, 4th Ed. Addison-Wesley Professional, 2009. 717 p.

[3] Кнут Д. Э. *Искусство програмирования. Т. 3. Сортировка и поиск*: пер. с анг. – 2е изд. – М.: Издательский дом «Вильямс», 2003. 832 с.

[4] Trello. [Електронний ресурс] – Режим доступу: https://trello.com/

[5] О. Д. Азаров, О. І. Черняк, та Л. А. Савицька, *Прикладне програмування*, Вінниця : ВНТУ, 2016. 131с.

[6] Т. М. Боровська, І.С. Колесник, та В.А. Северілов, *Метод оптимального агрегування в оптимізаційних задачах*Вінниця: УНІВЕРСУМ-Вінниця, 2009. – 229 с.

[7] Matt Telles, *Python Power! The Comprehensive Guide*, Thomson Course Technology, 2012. – 528 p.

Стаття надійшла: 28.08.2018

Відомості про авторів

Яровий Андрій Анатолієвич, д.т.н., проф., зав. кафедри комп'ютерних наук, ВНТУ, кафедра комп'ютерних наук.

Барабан Сергій Володимирович, к.т.н., старший викладач кафедри комп'ютерних наук, ВНТУ, кафедракомп'ютерних наук, Вінниця, Хмельницьке шосе, 95.

Криночкін Роман Володимирович, к.т.н., доцент, ВНТУ, кафедра радіотехніки, r@politex.org.ua, Вінниця, Хмельницьке шосе, 95.

A. A. Yarovyi[1], S. V. Baraban[1], R. V. Krynochkin[1]

INTELLECTUAL MODUL OF BROWSER MANAGEMENT SYSTEM OF IT-PROJECTS TRELLO

[1]Vinnytsia National Technical University

КОМП'ЮТЕРНІ СИСТЕМИ ТА КОМПОНЕНТИ

УДК 681.325.5

О. Д. Азаров[1], О. І. Черняк[1], О. Г. Муращенко[1]

МЕТОДИ ПЕРЕНЕСЕННЯ І ЗАПОЗИЧЕННЯ У ШВИДКОДІЮЧИХ ФІБОНАЧЧІЄВИХ ЛІЧИЛЬНИКАХ

[1]Вінницький національний технічний університет

Анотація. У даній статті описано підхід до організації перенесення при лічбі у модифікованій фібоначчієвій системі числення. Даний підхід полягає у тому, що на кожному такті лічби наряду з додаванням одиниці у молодший розряд в залежності від напрямку лічби виконується один із видів фібоначчієвого перетворення (F-перетворення) коду лічильника. Використання FL- та FR-перетворень дозволяє виконувати перенесення і запозичення ще до того, як виникне переповнення у молодших чи загублення значення у старших розрядах. Це дозволяє уникати ситуацій, при яких за один такт перенесення або запозичення розповсюджуються далі ніж через три розряди. У статті описано модифіковану фібоначчієву систему числення, наведено аналітичні вирази для опису базису і алфавіту та показано, як представляються у ній числа. Наведено аналітичні вирази, що описують FL- та FR-перетворення. Сформульовано твердження про те, що при виконанні всіх можливих фібоначчієвих перетворень на кожному такті лічби отриманий код буде мати не більше двох сусідніх одиниць. Це дозволяє організувати швидку лічбу за рахунок малого часу розповсюдження перенесення і запозичення.

Ключові слова: лічба, модифікована фібоначчієва система числення, фібоначчієве перетворення, перенесення, запозичення.

Аннотация. В данной статье описан подход к организации переноса при счете в модифицированной Фибоначчиева системе счисления. Данный подход заключается в том, что на каждом такте счета наряду с добавлением единицы в младший разряд в зависимости от направления счета выполняется один из видов Фибоначчиевого преобразования (F-преобразование) кода счетчика. Использование FL- и FR преобразований позволяет выполнять перенос и заимствование еще до того, как возникнет переполнение в младших или утеря значения в старших разрядах. Это позволяет избегать ситуаций, при которых за один такт перенос или заимствование распространяются дальше чем через три разряда. В статье описано модифицированную Фибоначчиева систему счисления, приведены аналитические выражения для описания базиса и алфавита и показано, как представляются в ней числа. Приведены аналитические выражения, описывающие FL- и FR преобразования. Сформулировано утверждение о том, что при выполнении всех возможных Фибоначчиевых преобразований на каждом такте счета полученный код будет иметь не более двух соседних единиц. Это позволяет организовать быстрый счет благодаря малому времени распространения переноса и заимствования.

Ключевые слова: счет, модифицированная фибоначчиевая система счисления, фибоначчиевое преобразование, перенос, заимствование.

Abstract. This article describes the approach to the organization of carry-over with the account in the modified Fibonacci numerical system. This approach is based on the fact that on each count cycle, along with the addition of a unit to the low order depending on the direction of the account, one of the Fibonacci transformation types (F-transformation) of the counter code is executed. Using FL- and FR-transformation allows you to carry out the carrying and borrowing even before there is an overflow in the lower or loss in the higher order bits. This makes it possible to avoid situations in which the carrying or borrowing is extended more than three orders in a single clock cycle. The article describes the modified Fibonacci numerical system, provides analytical expressions for describing the basis and the alphabet, and shows how the numbers are represented in it. Analytical expressions describing FL and FR transformations are given. An assertion is made that when all possible Fibonacci transformations are performed on each circle of count, the resulting code will have no more than two neighboring units. This allows you to organize a quick count due to the short transfer and borrowing time.

Keywords: counting, Fibonacci numerical system, Fibonacci transform, carrying, borrowing.
DOI: https://doi.org/10.31649/1999-9941-2018-42-2-55-63.

Вступ

Для широкого впровадження цифрових засобів, які у своїй структурі містять лічильники, потрібно щоб ці лічильники не лише мали високу швидкодію, але також забезпечували й інші параметри в залежності від характеру вирішуваних задач. Останнім часом при побудові засобів автоматизації і управління зростає популярність використання систем прямого цифрового синтезу (direct digital synthesis – DDS) аналогових сигналів за допомогою цифро-аналогових перетворювачів (ЦАП). Зокрема, широко використовуються DDS-системи для формування сигналів, що складаються з послідовності лінійно зростаючих, горизонтальних, а також лінійно спадаючих відрізків. У таких системах для формування цифрових кодів на вході ЦАП використовують швидкодіючі лічильники. Існують різні схемотехнічні способи підвищення швидкодії лічильників на основі структурних рішень. Проте, використання у DDS-системах традиційної двійкової системи числення породжує проблему "глітчів" – завад, які виникають на виході ЦАП під час перемикання розрядів. Причому, амплітуда цих "глітчів" напряму залежить від кількості розрядів, які перемикаються протягом одного такту. Відомо, що використання фібоначчієвих ЦАП дозволяє зменшити вплив таких завад [1]. Одним з важливих елементів DDS-системи з фібоначчієвим ЦАП є швидкодіючий лічильник, який повинен мати невелику кількість розрядів, що перемикаються протягом одного такту. Розробка такого лічильника є актуальною задачею.

Головним завданням на шляху вирішення даної задачі є зменшення довжини розповсюдження перенесення, що виникає на кожному такті лічби і обмежує її швидкість. Цього можна досягти за рахунок використання таких надлишкових систем числення, які мають адитивне співвідношення між вагами розрядів [2]. До цих систем числення належать також фібоначчієва система числення та система числення

золотої пропорції. У даних системах числення при додаванні можна виконувати перенесення раніше, ніж виникне переповнення [3-10].

У статті запропоновано принципи організації перенесення і запозичення при лічбі у модифікованій фібоначчієвій системі числення (МФ-системі числення) з метою зменшення довжини їх розповсюдження, що дасть можливість як підвищити швидкість лічби, так і зменшити завади на виході ЦАП.

Фібоначчієві перетворення в МФ-системі числення

Модифікована фібоначчієва система числення може бути описана за допомогою базису Ф і алфавіту D:

$$\left\{\begin{array}{l}\Phi:\{\varphi_0=1,\varphi_1=2,\underset{i>1}{\forall}(\varphi_i=\varphi_{i-1}+\varphi_{i-2})\}\\ D:\{0,1\}\end{array}\right\}, \quad (1)$$

де Ф – множина ваг розрядів φ_i, така, що $\varphi_0 = 1$, $\varphi_1 = 2$, а для кожного $i > 1$ $\varphi_i = \varphi_{i-1} + \varphi_{i-2}$;

D – множина з двох цифр 0 і 1.

В МФ-системі числення будь-яке ціле число X може бути представлене n-розрядним двійковим кодом $x_{n-1}x_{n-2}...x_{i}..x_1x_0$, де $x_i \in D$ відповідно до виразу (1), а n визначається за співвідношенням $\varphi_{n-1} \le X \le \varphi_{n-2}$. Позначемо n-розрядний двійковий код $x_{n-1}x_{n-2}...x_{i}..x_1x_0$ як X_0^n, а його частину довжиною в k розрядів, починаючи з i-го як X_i^k. В МФ-системі числення значення коду X_0^n визначається виразом:

$$X_0^n = \sum_{i=0}^{n-1} x_i \cdot w_i . \quad (2)$$

В МФ-системі числення між вагами розрядів існує фібоначчієве співвідношення (F-співвідношення):

$$F: \underset{i>1}{\forall}(\varphi_i = \varphi_{i-1} + \varphi_{i-2}) . \quad (3)$$

Для i-го розряду існує i-те F-співвідношення:

$$F_i: \varphi_i = \varphi_{i-1} + \varphi_{i-2} . \quad (4)$$

Фібоначчієве співвідношення дозволяє виконувати фібоначчієві перетворення кодів (F-перетворення). F-перетворення бувають двох типів: з перенесенням у старші розряди (FL-перетворення) і з перенесенням у молодші розряди розряди (FR-перетворення).

FL-перетворення коду X_0^n є умовною арифметичною операцією, що виконується над всіма його розрядами, крім нульового і першого. Дане перетворення полягає у тому, що для будь-якого i>1 у випадку, якщо $x_i=0$, $x_{i-1}=1$, $x_{i-2}=1$, виконується додавання одиниці в розряд x_i і віднімання одиниць у розрядах x_{i-1} та x_{i-2}:

$$FL(X_0^n) = \underset{x_i=0 \wedge x_{i-1}=1 \wedge x_{i-2}=1}{\forall} (X_0^n + \varphi_i - \varphi_{i-1} - \varphi_{i-2}) .$$

Відповідно до (4) i-те FL-перетворення коду записується виразом

$$FL_i(X_0^n) = \left\{\begin{array}{l} X_0^n + \varphi_i - \varphi_{i-1} - \varphi_{i-2} \text{ при } x_i = 0 \wedge x_{i-1} = 1 \wedge x_{i-2} = 1;\\ X_0^n \text{ при } x_i \neq 0 \vee x_{i-1} \neq 1 \vee x_{i-2} \neq 1; \end{array}\right\}.$$

FR-перетворення коду X_0^n також є умовною арифметичною операцією, що виконується над всіма його розрядами, крім нульового і першого. Дане перетворення полягає у тому, що для будь-якого i>1 у випадку, якщо $x_i=1$, $x_{i-1}=0$, $x_{i-2}=0$, виконується віднімання одиниці в розряді x_i і додавання одиниць у розряди x_{i-1} та x_{i-2}:

$$FR(X_0^n) = \underset{x_i=0 \wedge x_{i-1}=1 \wedge x_{i-2}=1}{\forall} (X_0^n - \varphi_i + \varphi_{i-1} + \varphi_{i-2}).$$

Відповідно, і-те FR-перетворення коду виконується над і-м, (і-1)-м та (і-2)-м розрядами цього коду і записується виразом

$$FR_i(X_0^n) = \begin{cases} X_0^n - \varphi_i + \varphi_{i-1} + \varphi_{i-2} \text{ при } x_i = 1 \wedge x_{i-1} = 0 \wedge x_{i-2} = 0; \\ X_0^n \text{ при } x_i \neq 1 \vee x_{i-1} \neq 0 \vee x_{i-2} \neq 0; \end{cases}.$$

FL- і FR-перетворення подібні до відомих операцій згортки і розгортки, що полягають у заміні одного коду на інший. Але, на відміну від операцій згортки і розгортки, що є логічними операціями, FL- і FR-перетворення є умовними арифметичними операціями додавання і віднімання, що виконуються над частинами коду. При цьому значення всього коду не змінюється, тому дані операції можна використовувати в якості перенесення і запозичення у процесі прямої або оберненої лічби. В МФ-системі числення перенесення і запозичення можуть виконуватись раніше, ніж виникне переповнення чи загублення значення у розрядах. Це дозволяє відокремити перенесення і запозичення від додавання чи віднімання одиниці при лічбі, завдяки чому вони мають обмежену довжину розповсюдження. Обмеженість довжини перенесення і запозичення покладено в основу побудови швидкодіючих лічильників в МФ-системі числення.

Перенесення при прямій лічбі в МФ-системі числення

Під час прямої лічби у даній системі числення на кожному такті над кодом лічильника, отриманим на попередньому такті, виконується FL-перетворення і до нього додається одиниця:

$$X_0^n(i) = FL(X_0^n(i-1)) + 1. \quad (5)$$

У випадку, якщо $(FL(X_0^n(i-1)))_0^3 = 011$, таке додавання призведе до перенесення з нульового у перший розряд, наприклад, 1001+1=1010. Якщо ж на попередньому такті $(X_0^n(i-1))_0^3 = 011$, то відповідно до (4)

$$(FL(X_0^n(i-1)))_0^3 = (FL_2(X_0^n(i-1)))_0^3 = 100.$$

Тому в цьому випадку після FL-перетворення попереднього коду додавання одиниці у його молодший розряд не призведе до перенесення у другий розряд, наприклад:

$$X_0^4(i-1) = 1011,$$
$$FL(1011)=FL_2(1011)=1100,$$
$$1100+1=1101.$$

Як видно з даного прикладу, після виконання і-го FL-перетворення розряди x_{i-1} та x_{i-2} мають нульові значення. Це дозволяє виконувати перенесення у дані розряди без його подальшого розповсюдження у старші розряди, тобто:

$$FL_i(X_0^n(i-1)) + 1 = (X_0^n(i-1))_{i+1}^{n-i-1} + X_0^{i+1}(i).$$

Збільшення будь-якого розряду коду лічильника, починаючи з другого, відбувається лише за рахунок FL-перетворення, тобто, перенесення з молодших розрядів. Очевидно, що при цьому вага таких перенесень завжди більша ваги молодшого розряду, на яку збільшується значення у лічильнику на кожному такті. Тому даний метод лічби не призведе до переповнення у розрядах коду лічильника. Більш того, кількість сусідніх одиниць коду, через які можливе перенесення на кожному такті, становить не більше двох. Це обґрунтовується наведеним далі твердженням.

Твердження 1. Якщо на кожному такті роботи фібоначчієвого лічильника додається одиниця до молодшого розряду та виконуються всі можливі FL-перетворення, то в його коді не може бути більше двох сусідніх одиниць, через які відбувається перенесення. Доведення даного твердження наведено в [11].

З твердження 1 слідує, що виконання всіх можливих FL-перетворень на кожному такті прямої лічби приводить до того, що перенесення у старші розряди не буде розповсюджуватись далі ніж через два розряди. Тому врахування його як паралельне перенесення потребує незначних апаратних витрат. Це дозволяє будувати в МФ-системі числення швидкодіючі лічильники з помірними апаратними витратами. Слід зазначити, що справедливість твердження 1 була доведена, виходячи з двох припущень: по-перше, що початковий код, з якого починається лічба, не містить більше двох сусідніх одиниць; по-друге, що при досягненні коду, у якому $(X_0^{n-1})_{n-2}^2 = 11$ (тобто, два старших розряди дорівнюють одиниці), подальша пряма лічба припиняється або лічильник встановлюється у початковий стан. У випадку, якщо виконується перше припущення, але не виконується друге, тобто, якщо при досягненні коду $110x_{n-4}...x_0$ лічба продовжується, то пряма лічба буде виконуватись коректно, але при цьому кількість сусідніх одиниць у коді лічильника з часом стане більшою двох. Це відбувається через те, що при досягненні коду $110x_{n-4}...x_0$ блокується виконання FL_{n-2}-перетворення, оскільки у всіх наступних тактах прямої лічби $x_{n-2}\neq 0$. На деякому наступному такті лічби це призведе до появи одиничного значення у розряді x_{n-3}, що, у свою чергу, заблокує виконання FL_{n-3}-перетворення, і так далі. Отже, продовження прямої лічби після досягнення фібоначчієвим лічильником коду $110x_{n-4}...x_0$ буде призводити до поступового збільшення кількості сусідніх одиниць у коді, починаючи зі старших розрядів. Тому на деякому k-у такті прямої лічби всі розряди коду лічильника матимуть одиничне значення:

$$\forall_{0\le i\le n-1} (x_i(k) = 1).$$

Відповідно до (3.2) і (3.1) значення даного коду $X(k)=\varphi_{n+1}-1$, де φ_{n+1} – (n+2)-е число Фібоначчі. Дане число визначає максимальну кількість одиниць, яку можна коректно підрахувати за допомогою n-розрядного лічильника. Тобто, $k=\varphi_{n+1}-1$. Подальша пряма лічба у даному лічильнику призведе до спотворення інформації у його коді. Тому в n-розрядному лічильнику після $\varphi_{n+1}-1$ тактів пряму лічбу потрібно припинити, або примусово скинути такий лічильник у початковий стан.

У випадку, якщо не виконується перше припущення, на якому базується доведення справедливості твердження 1, то це на будь-якому такті може призвести до неправильної роботи лічильника через виникнення у молодших розрядах переповнення, спричиненого невиконанням умови FL-перетворення у цих розрядах. Наприклад, встановлення лічильника у початковий код $X_0^{n-1}(0) = x_{n-1}...x_3 111$ вже на першому такті прямої лічби призведе до переповнення лічильника і спотворення результату через неможливість виконання FL_2-перетворення. Дана особливість прямої лічби в МФ-системі числення накладає обмеження на форму початкового коду у лічильнику, яке стосується кількості сусідніх одиниць у коді. Очевидно, що ці обмеження у першу чергу стосуються молодших розрядів, оскільки перенесення від додавання одиниці у нульовий розряд спочатку досягне їх. Тому визначимо обмеження, що накладаються на групу сусідніх одиниць у наймолодших розрядах.

Нехай у початковому коді лічильника молодші (k+1) розрядів дорівнюють нулю, d розрядів, починаючи з (k+1)-го, дорівнюють одиниці, а (k+d+1)-й розряд також дорівнює нулю, як це зображено на рисунку 1.

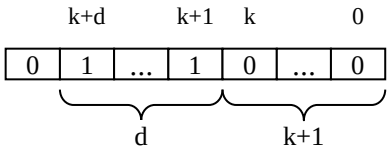


Рисунок 1 – Розташування сусідніх одиниць у d наймолодших розрядах лічильника

Тобто:

$$(x_{k+d+1} = 0) \wedge \forall_{k<i\le k+d} (x_i = 1) \wedge \forall_{0\le i\le k} (x_i = 0).$$

Визначимо кількість тактів, необхідну для того, щоб перенесення, яке виникає в режимі прямої лічби за виразом (5), розповсюдилось у k-й розряд. Очевидно, що дане перенесення не може виникнути раніше, ніж після φ_k тактів. У дійсності, воно виникає пізніше на деяку величину Δ_k. Отже, у режимі прямої лічби, організованої за виразом (3.5), перенесення у деякий k-й розряд коду надходить через $(\varphi_k+\Delta_k)$ тактів.

Знайдемо значення Δ_k. Особливістю МФ-системи числення є те, що для будь-якого парного k значення (φ_k-1) обчислюється за виразом:

$$\forall_{k_{mod2}=0}(\varphi_k - 1 = \sum_{i=1}^{k/2}\varphi_{2i}),$$

а для будь-якого непарного k це значення обчислюється за виразом:

$$\forall_{k_{mod2}=1}(\varphi_k - 1 = \sum_{i=0}^{(k+1)/2}\varphi_{2i}).$$

Тому такі значення представляються кодами, що у молодших розрядах мають $(k+k_{mod2})/2$ одиниць, розділених між собою нулями, наприклад:

$$\varphi_7\text{-}1=33_{(10)}=01010101_{(МФ)},$$
$$\varphi_{10}\text{-}1=143_{(10)}=01010101010_{(МФ)}.$$

Слід також вказати, що дана форма є єдиною для представлення кодів таких значень в МФ-системі числення. Внаслідок цього значення φ_k, що у режимі прямої лічби утворюється додаванням одиниці молодшого розряду до коду (φ_k-1), обчислюється за виразом

$$\forall_{k_{mod2}=0}(\varphi_k = \varphi_0 + \sum_{i=1}^{k/2}\varphi_{2i}),$$
$$\forall_{k_{mod2}=1}(\varphi_k = \varphi_1 + \sum_{i=2}^{(k-1)/2}\varphi_{2i}),$$

Отже, після φ_k тактів прямої лічби, починаючи з нуля, код у лічильнику буде мати у молодших розрядах $(k-k_{mod2})/2$ одиниць, розділених між собою нулями, після яких слідує ще одна одиниця, наприклад:

$$\varphi_7=34_{(10)}=01010110_{(МФ)},$$
$$\varphi_{10}=144_{(10)}=01010101011_{(МФ)}.$$

На наступних $(k-k_{mod2})/2$ тактах прямої лічби буде виконуватись розповсюдження перенесення у k-й розряд за рахунок виконання FL- перетворення, як це зображено на рисунку 2 для прикладу k=10. Тобто, $\Delta_k = (k-k_{mod2})/2$.

0	1	0	1	0	1	0	1	0	1	1	144-й такт
0	1	0	1	0	1	0	1	1	0	1	145-й такт
0	1	0	1	0	1	1	0	0	1	0	146-й такт
0	1	0	1	1	0	0	0	0	1	1	147-й такт
0	1	1	0	0	0	0	0	1	0	1	148-й такт
1	0	0	0	0	0	0	0	1	1	0	149-й такт

Рисунок 2 – Перенесення в 10-й розряд у режимі прямої лічби

Як видно з рисунку, починаючи з десятого такту, на кожному наступному такті за допомогою FL-перетворення відбувається розповсюдження перенесення з двох сусідніх розрядів. Тобто, розповсюдження перенесення через десять розрядів виконається за п'ять тактів. Таке перенесення повністю обнулить молодші десять розрядів коду, отриманого на десятому такті. Проте, протягом даного перенесення паралельно буде також відбуватись збільшення коду у молодших розрядах в результаті продовження прямої лічби. Це не вплине на розповсюдження перенесення, оскільки таке збільшення відбувається повільніше. Дійсно, на (k+i)-у такті в результаті FL-перетворення коду, отриманого на k-у такті, перенесення відбувається у 2i-й розряд, а перенесення, отримане за рахунок подальшої лічби розповсюджується в розряд з найменшою вагою, що більша чи дорівнює φ_i. Отже, у режимі прямої лічби, починаючи з нуля, перше перенесення в k-й розряд виникне через $N1_k$ тактів, де значення $N1_k$ обчислюється за формулою

$$N1_k = \varphi_k + (k - k_{mod2}) / 2.$$

Наступне перенесення у цей розряд виникне через $N2_k$ тактів, де $N2_k$ обчислюється за співвідношенням

$$N2_k = 2\varphi_k + (k - k_{mod2}) / 2.$$

Очевидно, що кількість сусідніх одиниць d у розрядах, починаючи з (k+1)-го повинна бути такою, щоб за цю кількість тактів в них виконались всі перенесення. Враховуючи, що при кожному виконанні FL-перетворення перенесення розповсюджується через два розряди, значення d повинно відповідати співвідношенню $d \le 2N2_k-1$, тобто

$$d \le 4\varphi_k + k - k_{mod2} -1. \quad (6)$$

На рисунку 3 зображено приклад максимальної кількості d сусідніх одиниць у молодших розрядах початкового коду при k = 1.

0	0	1	1	1	1	1	1	0	0	0-й такт
0	1	0	0	1	1	1	1	0	1	1-й такт
0	1	0	1	0	0	1	1	1	0	2-й такт
0	1	0	1	0	1	0	0	1	1	3-й такт
0	1	0	1	0	1	0	1	0	1	4-й такт

Рисунок 3 – Максимальна кількість сусідніх одиниць у молодших розрядах початкового коду при k = 1

У таблиці 1 для режиму прямої лічби подано значення максимальної кількості d сусідніх одиниць початкового коду лічильника, починаючи з k-го розряду при умові, що молодші k розрядів дорівнюють нулю.

Таблиця 1 – Максимальна кількість d сусідніх одиниць початкового коду, починаючи з k-го розряду при $(X(0)_0^{n-1})_0^{k-1} = 0$

k	φ_k	d
0	1	3
1	2	6
2	3	9
3	5	21
4	8	35
5	13	55
6	21	89
7	34	141

Значення d, наведені у таблиці 3.1, вказують на максимально допустиму кількість сусідніх одиниць, від k-го до (k+d-1)-го розрядів початкового коду лічильника при умові, що значення молодших k розрядів дорівнюють нулю. Якщо ж у k молодших розрядах знаходиться якесь початкове значення N(0), то враховуючи (6), d визначається за виразом:

$$d \le 4\varphi_k + k - k_{mod2} -1- 2(N(0) + N(0)_{mod2}). \quad (7)$$

Це дозволяє у режимі прямої лічби перевіряти на допустимість початковий код лічильника, починаючи з молодших розрядів. Наприклад, якщо встановлено початковий код 01111111110110, то відповідно до виразу (7) цей код є допустимим, оскільки для k = 0 виконується $d = 2 \le 3$, а для k = 3 виконується $\varphi_3 = 5$, $N(0) = 5$, $d = 9 \le 4 \cdot 5 + 3 - 1 - 1 - 2 \cdot (5 + 1)$. На рисунку 4 зображено процес прямої лічби, починаючи з даного коду.

0	1	1	1	1	1	1	1	1	1	0	1	1	0
1	0	0	1	1	1	1	1	1	1	1	0	0	1
1	0	1	0	0	1	1	1	1	1	1	0	1	0
1	0	1	0	1	0	0	1	1	1	1	0	1	1
1	0	1	0	1	0	1	0	0	1	1	1	0	1
1	0	1	0	1	0	1	0	1	0	0	1	1	0
1	0	1	0	1	0	1	0	1	0	1	0	0	1

0-й такт
1-й такт
2-й такт
3-й такт
4-й такт
5-й такт
6-й такт

Рисунок 4 – Розповсюдження перенесення у режимі прямої лічби, починаючи з коду 01111111110110

Запозичення при оберненій лічбі в МФ-системі числення

Робота у режимі оберненої лічби потребує виконання віднімання одиниці у наймолодшому розряді і запозичення зі старших розрядів. Під час оберненої лічби у МФ-системі числення на кожному такті над кодом лічильника, отриманим на попередньому такті, виконується FR-перетворення і від нього віднімається одиниця наймолодшого розряду:

$$X_0^{n-1}(i) = FR(X_0^{n-1}(i-1)) - 1. \qquad (8)$$

У випадку, якщо $(FR(X_0^{n-1}(i-1)))_0^2 = 100$, таке віднімання призведе до запозичення з другого у нульовий розряд, наприклад, 1100-1=1010. Проте, якщо на попередньому такті $(X_0^n(i-1))_0^2 = 100$, то відповідно до (3)

$$(FR(X_0^n(i-1)))_0^3 = (FR_2(X_0^n(i-1)))_0^2 = 011.$$

Тому в цьому випадку після виконання FR-перетворення над кодом, отриманим на попередньому такті, віднімання одиниці від значення наймолодшого розряду не призведе до запозичення з другого розряду, наприклад,

$$X_0^3(i-1) = 1100,$$
$$FL(1100)=FL_2(1100)=1011,$$
$$1011-1=1010.$$

Як видно з наведеного прикладу, після виконання k-го FR-перетворення над кодом, отриманим на попередньому такті, його розряди x_{k-1} та x_{k-2} мають одиничні значення. Це дозволяє виконувати запозичення з зазначених розрядів без його подальшого розповсюдження у старші розряди коду, тобто, виконується рівняння

$$FR_k(X_0^{n-1}(i-1)) - 1 = (X_0^{n-1}(i-1))_{k+1}^{n-k-2} + X_0^k(i).$$

При оберненій лічбі зменшення будь-якого розряду коду лічильника, починаючи з другого, відбувається лише за рахунок виконання FR-перетворення, тобто, запозичення зі старших розрядів. Очевидно, що при цьому вага таких запозичень завжди більша ваги наймолодшого розряду, на яку зменшується значення у лічильнику на кожному такті. Тому при наявності одиниць у старших розрядах коду даний метод лічби не призведе до переходу у від'ємне значення у молодших розрядах. Проте, даний висновок можна зробити лише відносно тих форм початкових кодів лічильника, що мають у своїх розрядах достатню кількість одиниць, наприклад, 0111111111. Існують такі форми початкових кодів, для яких неможливе коректне виконання оберненої лічби, наприклад, 10000000. Очевидно, що у цьому випадку вже на першому такті оберненої лічби відповідно до (8) молодший розряд перейде у від'ємне значення. Тому, як і для прямої лічби, для даного випадку постає задача знаходження допустимих форм початкових кодів лічильника. Для цього спочатку визначимо обмеження, що накладаються на кількість сусідніх нулів у такому коді.

Нехай у початковому коді лічильника X у молодших (k+1) розрядах міститься деякий код $(X_0^{n-1})_0^k$, d розрядів, починаючи з (k+1)-го, дорівнюють нулю, а (k+d+1)-й розряд також дорівнює одиниці, як це зображено на рисунку 5.

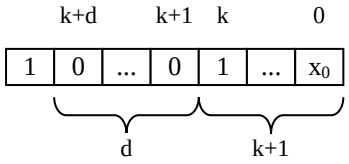


Рисунок 5 – Розташування сусідніх нулів у d розрядах лічильника

Тобто:

$$X = \varphi_{k+d+1} + \sum_{i=0}^{k} x_i \cdot \varphi_i .$$

Визначимо d. Очевидно, що для коректної оберненої лічби потрібно, щоб за X-φ_{k+d+1} тактів запозичення з (k+d+1)-го розряду досягло трьох наймолодших розрядів коду. Оскільки запозичення в МФ-системі числення реалізується за допомогою FR-перетворення, то на кожному окремому такті лічби воно розповсюджується на два розряди, як це зображено на рисунку 6.

1	0	0	0	0	0	0	0	0	0	1	0	0	1	0-й такт
0	1	1	0	0	0	0	0	0	0	0	1	1	0	1-й такт
0	1	0	1	1	0	0	0	0	0	0	1	0	1	2-й такт
0	1	0	1	0	1	1	0	0	0	0	1	0	0	3-й такт
0	1	0	1	0	1	0	1	1	0	0	0	1	0	4-й такт
0	1	0	1	0	1	0	1	0	1	1	0	0	1	5-й такт
0	1	0	1	0	1	0	1	0	1	0	1	1	0	6-й такт
0	1	0	1	0	1	0	1	0	1	0	1	0	1	7-й такт

Рисунок 6 – Розповсюдження перенесення у режимі оберненої лічби, починаючи з коду 10000000001001

Як видно з рисунку, для даного прикладу початкового коду на шостому такті запозичення з 13-го розряду досягає наймолодшої тріади розрядів. А в молодших розрядах початкового коду з 0-го по 3-й записано число 6. Тому за 6 тактів оберненої лічби воно стане дорівнювати нулю. Проте, завдяки запозиченню подальша лічба буде виконуватись коректно, що показано на рисунку у 7-у такті.

Отже, для коректного виконання оберненої лічби потрібно, щоб у початковому коді кількість сусідніх нулів d з (k+1)-го (k+d)-й розряди відповідала нерівності

$$d \le \begin{cases} -k + 2 \cdot \sum_{i=0}^{k} (x_i \cdot \varphi_i) \text{ при } (k+d)_{mod2} = 0, \\ 1 - k + 2 \cdot \sum_{i=0}^{k} (x_i \cdot \varphi_i) \text{ при } (k+d)_{mod2} = 1. \end{cases}$$

Так само, як і в режимі прямої лічби, вимоги до початкового коду у режимі оберненої лічби можуть бути легко виконані за допомого відповідного F-перетворення.

Проведені теоретичні дослідження перенесення у режимах прямої та оберненої лічби дозволяють розробити структурну організацію швидкодіючих лічильників у МФ-системі числення.

Висновки

1. Досліджено властивості модифікованої фібоначчієвої системи числення і описано фібоначчієві перетворення кодів, що можуть використовуватись при лічбі як перенесення і запозичення з малою довжиною розповсюдження.
2. Визначено обмеження, що накладаються на форми початкових кодів у режимах прямої та оберненої лічби.

3. Обґрунтовано, що при дотриманні обмежень на форми початкових кодів використання фібоначчієвих перетворень для виконання перенесення і запозичення дозволяє будувати швидкодіючі лічильники, в яких на кожному такті перенесення розповсюджується не далі, ніж на три розряди.

4. Використання таких лічильників у DDS-системах дозволить зменшити вплив "глітчів".

Список літератури

[1] Olexiy D. Azarov; Olexander G. Murashchenko, Olexander I. Chernyak, Andrzej Smolarz and Gulzhan Kashaganova "Method of glitch reduction in DAC with weight redundancy ", *Proc. SPIE* 9816, Optical Fibers and Their Applications 2015, 98161T (December 18, 2015); doi:10.1117/12.2229045; http://dx.doi.org/10.1117/12.2229045.

[2] О. Д. Азаров, О. І. Черняк, "Метод побудови швидкодіючих фібоначчієвих лічильників" *Проблеми інформатизації та управління* №2(46), с. 5-8, 2014.

[3] О. Д. Азаров, та О. І. Черняк, «Визначення довжини перенесення при додаванні в системах числення з адитивними та мультиплікативними співвідношеннями між вагами розрядів» *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*, Випуск 74, с. 401–407, 2004.

[4] О. Д. Азаров, та О. І. Черняк, «Структурна організація побітового множення і ділення кодів золотої пропорції» *Проблеми інформатизації та управління*, Вип. 3(21), с. 5–13, 2007.

[5] О. Д. Азаров, та О. І. Черняк, «Розрядність пристроїв порозрядного додавання в АМ-системах числення,» *Наукові праці Вінницького національного технічного університету*, [Електронний ресурс] № 4, 2010. Режим доступу : http://praci.vntu.edu.ua/index.php/praci/article/view/233.

[6] О. Д. Азаров, та О. І. Черняк, «Структурна організація побітового додавання і віднімання кодів золотої 1-пропорції з урахуванням знаків,» *Інформаційні технології та комп'ютерна інженерія*, № 3(22), с. 13–16, 2011.

[7] О. Д. Азаров, та О. І. Черняк, «Аналіз витрат обладнання пристроїв побітової арифметики у системі числення золотої 1-пропорції,» *Проблеми інформатизації та управління*, Київ: НАУ, № 2(38), с. 5-9, 2012.

[8] О. Д. Азаров, О. І. Черняк, *Повнофункціональна побітова потокова арифметика зі зменшеними витратами обладнання*, Вінниця Україна: ВНТУ, 2013, 200с.

[9] О. Д. Азаров, та О. І. Черняк «Обмеження адитивних співвідношень при порозрядній потоковій обробці в АМ-системах числення,» *Інформаційні технології та комп'ютерна інженерія*. № 3(31), с. 67-71, 2014.

[10] О. Д. Азаров, О. І. Черняк, та О. Г. Муращенко, «Порозрядне додавання в АМ-системах числення на основі адитивних перетворень,» *Проблеми інформатизації та управління*. № 1(45), с. 14-21, 2014.

[11] О. Д. Азаров, О. І. Черняк, та О. Г. Муращенко, «Інформаційні аспекти лічби у модифікованій фібоначчієвій системі числення,» *Інформаційні технології та комп'ютерна інженерія*. № 1(38). - с. 48-52, 2017.

Стаття надійшла: 28.08.2018.

Відомості про авторів

Азаров Олексій Дмитрович, д. т. н., професор, декан факультету інформаційних технологій та комп'ютерної інженерії Вінницького національного технічного університету, заслужений працівник освіти України.

Черняк Олександр Іванович, к.т.н., доцент кафедри обчислювальної техніки Вінницького національного технічного університету.

Муращенко Олександр Геннадійович, інженер ТОВ "Він-Інтерактив"; адреса: 21021 м. Вінниця.

O. D. Azarov[1], O. I. Chernyak[1], O. G. Muraschenko[1]

THE TRANSFER AND BORROWING METHODS IN FAST FIBONACCI COUNTERS

[1]Vinnitsa National Technical University

ДО ВІДОМА АВТОРІВ

Найновіші правила оформлення і подання статей знаходяться на сайті журналу
http://itce.vntu.edu.ua/