DOI 10.31891/2307-5732-2022-309-3-221-228
УДК 637.5.02

ПРИГОЖЕВ О. С.
Державний університет «Одеська політехніка»
https://orcid.org/0000-0001-8532-8897
e-mail: o.s.prygozhev@op.edu.ua

МОВНОНЕЗАЛЕЖНИЙ РЕПОЗИТАРІЙ ПРОГРАМНОГО КОДУ

У статті .розглядається побудова мовнонезалежного репозитарію програмного коду, який засновано на новій моделі представлення програмного коду на основі мультиграфу із токенами предметної області . Розглянуто основні аналоги системи. Наведено порівняльний аналіз аналогічних рішень. На основі аналізу зроблено висновок про необхідність розробки нової моделі представлення програмного коду на основі мультиграфу. Розроблена модель представлення оброблюваних даних для такої моделі – токен предметної області. Розроблено представлення основних алгоритмічних конструкцій на основі розробленої моделі. Синтезовано архітектуру репозитарію, засновану на новій моделі предметної області

Keywords: multigraph, tokens of subject area, program code, software code repository

Oleksandr PRYGOZHEV
State University “Odessa Politechnic”

LANGUAGE INDEPENDENT SOFTWARE CODE REPOSITORY

The article considers the construction of a language-independent repository of program code, which is based on a new model of program code representation based on a multigraph with tokens of the subject area. The main analogues of the system and the models on which they are based are considered: finite state machines, Petri nets, P-schemes. A comparative analysis of similar solutions is given. During the analysis, three main indicators were used: the ability to model the implementation of the program step by step ("white box"), the ability to build an interface in natural language, isomorphism of basic mathematical concepts. Based on the analysis, it was concluded that it is necessary to develop a new model of program code based on multigraph, which will represent the functioning of the program as a "white box", be able to build an interface in natural language and be a completely isomorphic basic mathematical concept. The model of representation of the processed data for such model - the token of subject area is developed. It includes information about the attribute name, its type, and the current value. The model code based on the multigraph is multilayered. Each layer reproduces some program code procedure. Each layer contains a representation of the algorithm of the procedure. Representation of the basic algorithmic constructions is developed: compositions of two operators, alternatives and a cycle with postcondition and precondition on the basis of the developed multigraph model. Data processing is modeled as the movement of one subject area token by a synthesized multigraph. The repository architecture based on the new subject area model is synthesized

Keywords: multigraph, tokens of subject area, program code, software code repository

Постановка проблеми у загальному вигляді та її зв’язок із важливими науковими чи практичними завданнями

Сучасний етап розвитку програмних систем нерозривно пов’язаний із гнучкими методологіями розробки програмного забезпечення, зокрема такими як Agile, Scrum та інші подібні методології. Ці методології передбачають участь замовника у процесі розробки програмного забезпечення та швидке представлення перших версій (прототипів) програмного продукту на його розгляд..

У таких умовах важливою складовою програмного продукту є інструментальні засоби контролю версій, такі як GitLab , GitHub або Bitbucket. В основі цих продуктів полягає поняття гілок версій, які створюються розробником під час процесу розробки. До кожної з гілки може бути прикріплене символьне ім’я та опис, що дозволяє визначити, до якого етапу розробки належить та чи інша робоча версія програмного продукту.

Кожна з гілок у цих репозитаріях містить увесь код програмного продукту у стиснутому вигляді, щоб під час відкату до попередніх версій можливо було відтворити увесь стан програмного коду продукту.

Недоліком цих систем є зберігання коду програми у вигляді текстового файлу, що дозволяє відтворити версію вирішення задачі лише на тій мові програмування, на який написаний продукт. У той же час, якщо така сама задача постає у іншому проекті на іншій мові програмування, її необхідно повторно розробити.

Цю проблему досить часто вирішують за допомогою компонентного підходу, коли код вирішення задачі може бути динамічно прилінкований до програмного продукту. Однак цей підхід спрацьовує лише коли розроблюваний продукт буде працювати в тому самому компонентному середовищі, для якого була створена бібілотека динамічного компонування.

Таким чином, актуальним є створення моделей та методів, що дозволяють зберігати рішення задачі у незалежній від мови програмування формі представлення та повторно використовувати рішення задачі у різних мовах програмування для підвищення продуктивності розробки програмного забезпечення

Для вирішення цієї задачі перспективним є використання моделей подання знань для представлення програмного коду. Розглянемо більш детально існуючі моделі з метою обрання найбільш перспективної для вирішення цієї задачі

Аналіз досліджень та публікацій

До сучасних моделей подання знань, за допомогою яких можливо представити знання про процеси, відносяться, здебільшого, графові моделі: мережі Петрі [1,2], кінцеві автомати [3,4], Р-схеми[5]

Мережа Петрі являє собою дводольний мультиграф, у якому циркулюють відповідні мітки. Виконання дії ілюструється за допомогою вершин переходів, існування умов визначається за допомогою вершин позицій. У позиціях існують мітки, які дозволяють наочно представити процес обробки даних, зокрема, у паралельних системах. Це дозволяє застосовувати їх не лише у паралельних системах, а й під час тестування різноманітних систем

Опис дій на природній мові можливий шляхом зіставлення пояснень ідентифікаторам вершин. Ці пояснення та їх зміст можуть бути у формі довільного тексту, що значно ускладнює процес застосування їх у природно-мовному інтерфейсі

До недоліків мереж Петрі слід віднести її неізоморфність базовим математичним поняттям. Відомо, що поняття n-арної операції визначається як $n+1$-арне відношення. Зазвичай відношення позначають навантаженими або ненавантаженими ребрами графу. У той же час, у мережі Петрі дія визначається за допомогою вершини, тобто елементу множини.

Кінцеві автомати є більш ізоморфною до цих понять схемою, оскільки переходи, зазвичай, позначаються ребрами, що відповідає поняттю операції, а стан позначається вершинами, що відповідає поняттю елемента множини. Кінцеві автомати не мають засобів, які могли б зіставити їм природно-мовний інтерфейс. Так само ускладнене й формулювання основних алгоритмічних фрагментів.

Р-схеми, представляють обчислювальний процес у формі послідовності ребер, кожне з яких має умову та дію, яка виконується при настанні умови. Поняття дії, як зазначалося вище, пов’язане з поняттям відношення, а отже, ребра графу. Таким чином Р-схема ізоморфна поняттям дії та елементу множини.

Водночас, у Р-схемах відсутні засоби демонстрації обчислювальних процесів, на кшалт множини міток у мережах Петрі. Також самі операції не завжди визначені математично. Досить часто вони виражені словами, інтерпретацію змісту яких покладається повністю на людину. Цей недолік є схожим із таким самим недоліком у мереж Петрі.

Усі розглянуті нами моделі зведемо у таблицю. Символ «+» у таблиці позначає виконання вимоги, «-» - невиконання вимоги

Таблиця 1

Порівняльна характеристика моделей, орієнтованих на процеси

	«Білий ящик»	Природномовність	Ізоморфність
Мережі Петрі	+	-	-
Кінцеві автомати	+	-	-
Р-схеми	+	-	+

Метою роботи є розробка моделі програмного коду на основі графу з метою його використання як внутрішньої форми представлення для репозитарію програмного коду

Для досягнення цієї мети необхідно вирішити наступні задачі: розробка моделі представлення даних для моделі програмного коду; синтез мовнезалежної моделі програмного коду; синтез програмної системи на основі розробленої мовнонезалежної моделі.

Розробка моделі представлення даних для моделі програмного коду

Найменшою одиницею предметної області є атрибут, тип якого є простим типом даних. Формально множину таких атрибутів предметної області будемо визначати за наступною формулою:

$$Attr = Id \times \mathrm{P}\left(Type\right) \times Type \qquad (1)$$

де $\mathrm{I}d \subset \mathrm{Z}_{+} \bigcup \{0\}$ - множина семантичних чисел деякого атрибуту, Type – множина значень типу даних. $\mathrm{P}\left(Type\right)$ – булеан множини значень типу даних.

Таким чином, вектор $\mathrm{a}ttr \in Attr$ містить повний опис усіх можливих змінних будь-яких типів даних. При цьому опис типу, саме як множини значень, дозволяє досить широко формулювати типи атрибутів. Але для множини $Type$ завжди виконується співвідношення $Type \subset \mathrm{Z}$. Знак суворого включення означає, що тип, у загальному випадку, не охоплює всієї множини цілих чисел. При цьому, якщо тип символьний, під цілим числом мається на увазі код символу у кодовій таблиці

Для моделювання таких структур даних, як масиви, формулу (1) необхідно розширити, включивши у неї, відповідно індекс масиву:

$$Attr = Id \times Index \times \mathrm{P}\left(Type\right) \times Type \qquad (2)$$

де $\mathrm{I}d \subset \mathrm{Z}_{+} \bigcup \{0\}$ - множина семантичних чисел деякого атрибуту, $Type$ – множина значень типу даних. $\mathrm{P}\left(Type\right)$ – булеан множини значень типу даних, $Index = \{i : i \in \mathrm{Z}\,\mathrm{та}\,i \geq 0\}$ – множина можливих значень індексів для масивів.

Семантичні числа визначаються наступною формулою

$$\begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} * \begin{pmatrix} const_{ctx}^{0} \\ \vdots \\ const_{ctx}^{i-1} \end{pmatrix} \quad (3)$$

де $c_1,...,c_n$ – цілі числа з множини цілих чисел від 1 до n, що відповідають символам, які використовуються під час формулювання елементів множини Id. $const_{ctx}$ – константа контексту, яка встановлюється рівною $\mathrm{n}+1$ та відіграє роль ваги розряду системи числення.

Фактично, семантичне число являє собою десятковий еквівалент слова, якщо його представляти як число у системі числення. Семантичне число «0» позначає константне значення деякого типу даних, без якогось його тлумачення. Прикладом такого випадку є, наприклад число 3,14 без додаткового зазначення, що це наближення константи «пі».

Сформулюємо логіко-функціональну модель атрибуту. Вимогою до семантичного числа атрибуту є додатність цього числа. Введемо у розгляд одномісний предикат P_0, який має наступну область істинності: $P_0 = \{(id,i,T,t) : id \in Id\, ma\, id > 0\}$. Це визначення базується на векторі $(id,i,A,t) \in Attr$, де $id \in Id$, $i \in Index$, T – це множина значень типу даних, $t \in T$ – конкретне значення типу.

Вимогу щодо індексу декартового добутку атрибутів визначимо за допомогою предикату P_{index}, область істинності якого визначається на множині $\mathrm{A}ttr$. наступним чином $P_{index} = \{(id,i,A,t) : i \in Z \text{ та } i \geq 0\}$, де $(id,i,A,t) \in Attr$.

Введемо у розгляд предикат для визначення того факту, що тип даних заданий. Це означає, що третій компонент вектору не є пустою множиною. Це формулюється за допомогою предикату P_{type} із наступною областю істинності: $P_{type} = \{(id,i,A,t) : A \neq \varnothing\}$. Звернемо увагу, що при представленні за допомогою цієї моделі нетипізованих мов програмування ця множина не буде пустою. Множина A буде визначатися, як множина усіх можливих значень змінної. Вимога до четвертого компоненту вектора, яка полає у приналежності значення множині, яка є третьою компонентою, сформулюємо у вигляді предикату $P_t = \{(id,i,A,t) : t \in A\}$

Додатково визначимо двомісний предикат рівності семантичних чисел атрибутів, який визначається за наступною формулою: $P_{sem} : Attr^2 \to \{0;1\}$. Область істинності цього предикату має наступну характеристичну властивість: $\{(attr_1; attr_2) : np_1 attr_1 = np_1 attr_2\}$. За аналогією визначимо двомісні предикати рівності індексів (P_{eqi}), типів (P_{eqt}) та значень (P_{eqv}). Єдина відмінність від визначення предикату P_{sem} полягає в тому, що для P_{eqi} використовується друга проекція вектору атрибута, для P_{eqt} – третя, для P_{eqi} – четверта.

Таким чином, сформульована наступна логіко-функціональна модель для задачі

$$\mathrm{M}_{attr} = (Attr, \{P_0, P_{index}, P_{type}, P_t, P_{eqi}, P_{eqt}, P_{eqv}\}) \quad (4)$$

Сигнатуру предикатів цієї логіко-функціональної моделі будемо позначати P.

Оскільки поняття змінної співпадає з поняттям одно елементного масиву з мінімальним значенням індексу, надалі для опису атрибуту будемо використовувати формулу (2).

Надалі введемо у розгляд багатоосновну алгебру атрибутів . Основами цієї алгебри будуть множина $Attr$ та множина цілих чисел. Визначимо операції на цій алгебрі

Додаванням значення атрибутів будемо називати операцію $\varphi_a : Attr^2 \to Attr$. Оскільки цій бінарній операції можна поставити у відповідність тернарне відношення, із наступною породжуючою функцією:

$$\{(attr_1, attr_2, attr_3) : np_4 attr_3 = np_4 attr_2 + np_4 attr_1\} \quad (5)$$

де $attr_1, attr_2, attr_3 \in Attr$ – елементи множини елементарних атрибутів, np_4 – взяття четвертої проекції цих векторів.

Таким чином, формула (5) задає багатовимірну таблицю Келі для операції φ_a.

Визначимо також функцію додавання константи до значення атрибуту $\varphi_{ac} : Attr \times \mathrm{Z} \to Attr$, яка визначається за наступною формулою

$$\left\{\left(attr_1, z, attr_2\right) : \text{пр}_4 attr_2 = \text{пр}_4 attr_1 + \text{z}\right\} \quad (6)$$

За аналогією з операцією суми атрибутів φ_a визначимо операцію множення атрибутів $\varphi_m : Attr^2 \to Attr$

$$\left\{\left(\text{attr}_1, \text{attr}_2, \text{attr}_3\right) : \text{пр}_4 \text{attr}_3 = \text{пр}_4 \text{attr}_2 * \text{пр}_4 \text{attr}_1\right\} \quad (7)$$

де $attr_1, attr_2, attr_3 \in Attr$ – елементи множини елементарних атрибутів, np_4 – взяття четвертої проекції цих векторів.

Введемо у розгляд операцію множення на константу значення атрибуту $\varphi_{mc} : Attr \times \text{Z} \to Attr$. Таблиця Келі для неї буде визначатися за наступною характеристичною властивістю:

$$\left\{\left(attr_1, z, attr_2\right) : \text{пр}_4 attr_2 = \text{пр}_4 attr_1 * \text{z}\right\} \quad (8)$$

Також введемо у розгляд операцію ототожнювання індексу. Функція виду $f : Attr \times \text{Z} \to Attr$, яка характеризує ототожнювання індексу, задається наступною характеристичною властивістю:

$$\left\{\left(attr_1, z, attr_2\right) : \text{пр}_1 attr_1 = \text{пр}_1 attr_2 \ \& \text{пр}_2 attr_2 = z\right\} \quad (9)$$

Операцію зміщення індексу $g : Attr \times \text{Z} \to Attr$ задається за допомогою наступної відповідності

$$\left\{\left(attr_1, z, attr_2\right) : \text{пр}_1 attr_1 = \text{пр}_1 attr_2 \ \& \text{пр}_2 attr_2 = \text{пр}_2 \text{attr}_1 + z\right\} \quad (10)$$

Отже, оскільки тип даних є, загалом, абстрактною багатоосновною алгеброю задамо багатоосновну алгебру атрибутів наступним чином

$$\text{A}_{\square\square\square\square} = \left(Attr, \text{Z}, \{\varphi_a, \varphi_m, \varphi_{mc}, \varphi_{ac}, f, g\}\right) \quad (11)$$

де $Attr$ – множина атрибутів предметної області, Z - множина цілих чисел, φ_0 – операція додавання значень атрибутів, φ_m – операція множення значень атрибутів, f – операція ототожнювання індексу, g – операція зміщення індексу.

Операції, які вказані у сигнатурі алгебри (6) будемо вважати базовими операціями у мовнонезалежній моделі програмного коду. Сама алгебра (6) описує, з точки зору програмування, тип даних який будемо називати токен предметної області.

Надалі, для опису дій потрібно розширити багатоосновну алгебру (9) теоретико-множинними операціями над булеаном атрибутів. Тоді ця багатоосновна алгебра матиме наступний вигляд:

$$\text{A}_{\square\square\square\square} = \left(Attr, \text{Z}, \text{P}\left(Attr\right); \{\varphi_a, \varphi_m, \varphi_{mc}, \varphi_{ac}, f, g, \setminus, \cup\}\right) \quad (12)$$

Теоретико множинні операції об'єднання та різниці у цьому визначенні застосовуються до булеану атрибутів. Сигнатуру операцій цієї багатоосновної алгебри складають операції алгебри (9), а також теоретико-множинні операції об'єднання та різниці.

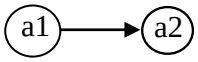
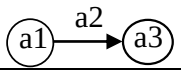
Багатоосновна алгебра (10) та логіко-функціональна модель (4) поєднаємо у багатоосновну алгебраїчну систему, в якій у якості основи буде використано основу алгебри (10), в якості сигнатури операцій виступає сигнатура алгебри (10), а в якості сигнатури предикатів – сигнатура логіко-функціональної моделі (4), яка додатково розширена предикатом рівності множин, який визначено на булеані $\text{P}\left(Attr\right)$

Розробка мовнонезалежної моделі представлення програмного коду

Відомо, що будь-яка n-арна операція може бути описана n+1-арним відношенням. Зокрема, унарна операція описується бінарним відношенням, а бінарна операція – відношенням арності три. У всіх випадках, остання компонента вектора є результатом операції.

Таблиця 2

Правила представлення операцій у вигляді графу

Операція	Тип відношення	Елемент у відношенні	Вид ребра графу
Унарна	Бінарне	$\left(a_1; a_2\right)$	
Бінарна	Тернарне	$(a_1; a_2; a_3)$	

З іншого боку бінарне відношення може бути представлене, як ребро графу, яка поєднує дві вершини, а бінарну операцію можна представити як навантажене ребро графу. Правила цього представлення, що надалі використовуються наведені у таблиці 2

Також, кожному ребру зіставимо у відповідність тип ребра, значення якого вибирається з множини, що є об'єднанням сигнатури операцій та сигнатури предикатів алгебраїчної системи (10). Фактично,

сигнатура предикатів та сигнатура операцій є множиною можливих типів ребер. Отже маємо надалі розглядати навантажений мультиграф.

Кожну вершину пропонованого мультиграфу опишемо наступним рівнянням:

$$\mathrm{V} = \mathrm{I} \times \mathrm{P}\,(\mathrm{Attr}) \tag{13}$$

де $I \subseteq \mathrm{Z}$ – ідентифікатор вершини (ціле число), $\mathrm{P}\,(\mathrm{Attr})$ – усі можливі набори токенів предметної області, які визначаються за формулою (2).

Вектор $v \in V$ складається з двох компонентів. Перша проекція – номер вершини, друга містить набір значень для предметної області, або пусту множину

Множина V відіграє роль універсальної множини для вершин графів. Ідентифікатор вершини грає роль постійної унікальної константи для цієї вершини та дозволяє зробити структуру графу незмінною.

Універсальна множина ребер цього мультиграфу описується за формулою:

$$E = V^2 \times Attr \times Type \tag{14}$$

Де V – множина вершин графу, яка визначається формулою (13). Множина $Attr$ – множина атрибутів, $Type$ – тип вершини, $Type = P \bigcup \Omega$.

Для подальшого вирішення задачі введемо у розгляд алгебру Дейкстри [6], яка у якості основ містить сигнатуру предикатів та операцій, які визначено логіко-функціональній моделі (4) та у багатоосновній алгебрі (10), а як сигнатуру операцій відомі алгоритмічні операції композиції, альтернативи та циклу. Розглянемо побудову операцій композиції, альтернативи та циклу у вигляді такого мультиграфу.

Композиція операцій у алгебрі Дейкстри є узагальненням відомої операції композиції на випадок цієї алгебри. Вона позначає послідовне виконання операцій без додаткових умов. Результат першої операції є вхідними даними для другої операції. Приклад визначення двох операцій на графі наведено на рис.1

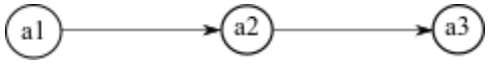


Рис. 1. Представлення послідовного виконання операцій

На рис. 1 позначено граф, який містить три вершини $a_1, a_2, a_3 \in V$. Індекси позначають перший компонент вектору. Другий компонент – пуста множина. Обидва ребра мають тип «операція» (тип Ω). Присвоєння ребрам номерів відбувається за будь-яким довільним алгоритмом обходу графа.

Наступним елементом є елемент умови, та частковий його випадок – цикл. Будь-яка умовна дія складається з перевірки умови (ребро має тип P) та одразу після нього виконання однієї чи декількох дій (послідовність ребер типу Ω). Приклад опису найпростішого умовного оператору на графі наведений на рис 2

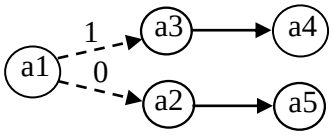


Рис. 2. Представлення умови на графі

Пунктиром позначено ребра типу P, суцільною лінією – ребра типу Ω. Символ «1» над ребром типу P позначає істинне значення предикату, символ «0» - хибне значення предикату. Нумерація здійснюється так само, як і у випадку послідовного виконання операцій.

Представлення циклів на цьому графі відбувається аналогічно умовному оператору. У випадку циклу з постумовою, предикатне ребро, яке відповідає хибному оператору одночасно здійснює повернення до вершини, з якої починається операторне ребро початку циклу(рис.3)

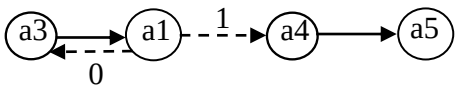


Рис. 3 Представлення циклу на графі

У випадку циклу з передумовою, ребро, яке відповідає останньому оператору циклу здійснює повернення на ребро визначення умови, а хибне ребро розпочинає цикл. Вихід з циклу здійснюється за допомогою істинного предикатного ребра (рис.4)

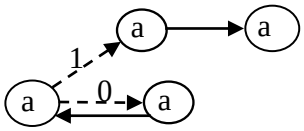


Рис. 4 Представлення циклу із передумовою

Представлення всіх алгебраїчних записів базових алгоритмічних фрагментів на графі дозволяє формально представити за допомогою методів трансляції, будь який код процедури у мові програмування в уніфікованому вигляді, незалежно від синтаксичних особливостей мови програмування. Але існує проблема представлення програмного коду, у якому є багато різних процедур, а також представлення взаємодії цих процедур. У алгебро алгоритмічному підході це вирішується за допомогою клонів – алгебр, що містять у якості основи операції деякої алгебри, а у якості сигнатури – єдину операцію суперпозиції (підстановки) операцій. З точки зору коду програмного забезпечення, суперпозиція є викликом процедури програмного коду.

Введемо у розгляд поняття початкової вершини, як вершини, з якої виходить ребро, що відповідає першому операторному виклику у програмі. Так само, заключною вершиною буде вершина, до якої входить ребро заключного оператора чи операції програми.

Враховуючи це, операція суперпозиції може бути представлена за допомогою введення графу шарів. Кожному шару у цьому графі однозначно відповідає граф, який побудовано для схем алгебри Дейкстри. До ребер графу шарів пред'являються наступні вимоги: одне ребро поєднує довільну вершину на одному шарі зі стартовою вершиною, друге ребро поєднує заключну вершину одного шару з довільним ребром на іншому шарі. Слід зазначити, що, враховуючи зміст та аргументи операції суперпозиції, ці ребра є окремим типом ребер – ребра-виклику. Приклад такого графу наведено на рис. 5

На ребрах, які поєднують різні шари може бути визначене введення локальних змінних за допомогою теоретико-множинних операцій об'єднання та різниці. Першим аргументом цих операцій є друга проекція початкової вершини ребра, яке з'єднує шари. Другим аргументом є множина локально визначених атрибутів, які будуть використані під час виконання дії.

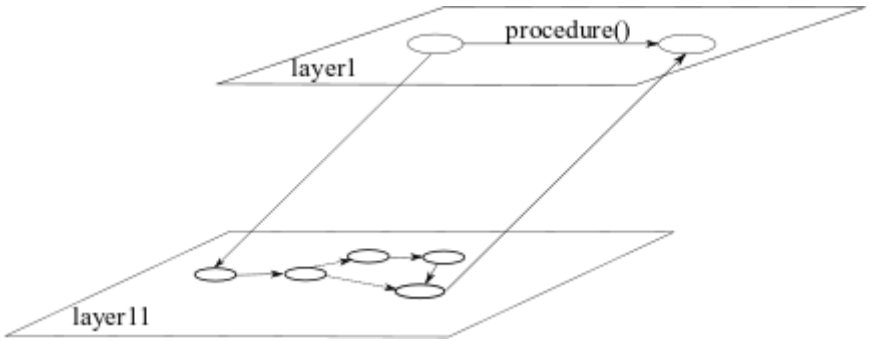


Рис. 5 Представлення процедури

Якщо, наприклад, процедура (див. рис 5) має декілька реалізацій, то вони розташовуються на окремих дочірніх шарах, які безпосередньо пов'язані із layer1. В цьому випадку для дочірніх шарів можуть зазначатися окремі ідентифікатори та додаткова інформація (наприклад, частота викликів саме цієї реалізації). Слід зазначити, що мультишарове представлення можливе лише для операторних ребер. При цьому вершини, які поєднує ребро мають послідовні номери.

Побудоване представлення у вигляді графу дозволяє вирішити ще одну важливу проблему, яка існує у моделі. У векторі атрибутів немає можливості представити дробові числа. Це пов'язано із тим, що дробові числа у сучасних обчислювальних системах утворюються за допомогою внутрішніх алгоритмів обробки цілочисельних масивів даних. Застосування багатошарової моделі (див. рис. 6) дозволяє представити дробові числа саме у тому вигляді, у якому вони обробляються у обчислювальній системі. У цьому випадку процедурою є виконання будь-яких арифметичних дій над цими числами. Представлення символів у обчислювальній системі є так само алгоритмом, який ставить у відповідність цілому числу зображення відповідного символу. Цей алгоритм так само може бути побудований за допомогою багатошарового представлення (див. рис. 6)

Побудова мовнонезалежного репозиторію програмного коду

Запропонована модель у вигляді багатошарового графу дозволяє побудувати мовнонезалежний репозитарій програмного коду. Він буде базуватися на представленні програмного коду у вигляді розробленого багатошарового графу.

Діаграма пакетів для представленого репозитарію наведено на рис.6 .

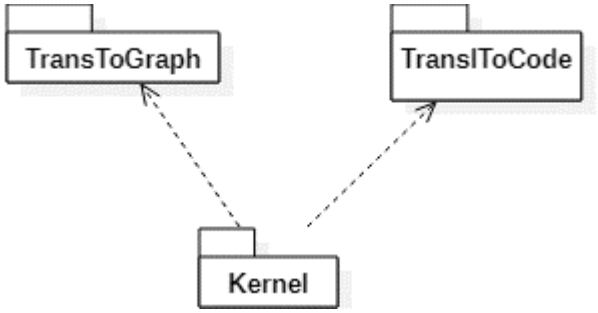


Рис. 6 Діаграма пакетів для мовнонезалежного репозитарію програмного коду

Модуль TransToGraph призначениий для побудови транслятора представлення програмного коду у графову форму представлення. Блок TranslToCode призначений для переведення графової форми представлення у представлення у вигляді програмного коду. Блок Kernel призначений для виконання операцій з імітаційного моделювання на розробленій графовій моделі.

Діаграма класів для модуля Kernel наведено на рис. 7

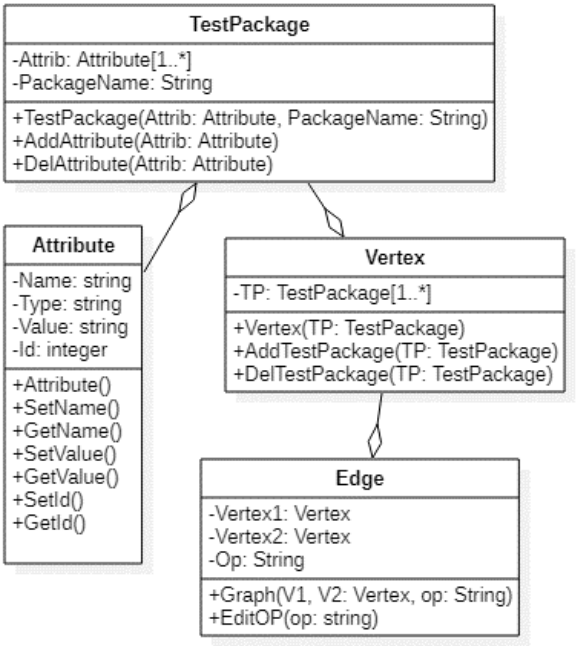


Рис. 7. Діаграма класів для модуля kernel

Модуль Kernel складається з класів Attribute, TestPackage, Vertex, Edge. Клас Attribute представляє собою атрибут предметної області, що моделюється. Ці атрибути об'єднуються у тест-пакети, які представлені класом TestPackage. Відповідно до розглянутої моделі тест-пакети знаходяться у вершині графу, що показано на діаграмі класів як об'єкт класу Vertex. Вершини з'єднуються між собою ребрами різних типів, які представлені як клас Edge

Модулі TransToGraph та TranslToCode призначені відповідно для переводу коду у представлення у вигляді графу і представлення графу у відповідний фрагмент програмного коду. Трансляція графу у програмний код здійснюється за допомогою алгоритмів обходу графу.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

У роботі розглянуто побудову моделі для репозитарію програмного коду, який дозволяє зберігати рішення задач у мовнонезалежному форматі графу та додавати до рішення назви процедур на природній мові.

У статті побудовано мовнонезалежну модель представлення програмного коду у вигляді графу. Побудовано математичне представлення для природномовного інтерфейсу репозитарія. Використання цього репозитарія дозволяє систематизувати рішення задач та дозволити автоматизовано побудувати програмний код для рішення задачі

Література

1. Осколков А. П. Сети Петри – инструмент для описания и исследования динамических систем [Електронний ресурс] / А. П. Осколков // Системи обробки інформації. - 2014. - Вип. 6. - С. 149-151. - Режим доступу: http://nbuv.gov.ua/UJRN/soi_2014_6_37

2. Шостак И. В. Метод расширения модели логистической цепи поставок, представленной в форме двухуровневой вложенной сети Петри [Електронний ресурс] / И. В. Шостак, Я. Рахими // Радіоелектронні і комп'ютерні системи. - 2019. - № 1. - С. 82–90. - Режим доступу: http://nbuv.gov.ua/UJRN/recs_2019_1_11

3. Крывый С. Л. Конечные автоматы в информационных технологиях [Електронний ресурс] / С. Л. Крывый // Кибернетика и системный анализ. - 2011. - Т. 47, № 5. - С. 3-20. - Режим доступу: http://nbuv.gov.ua/UJRN/KSA_2011_47_5_3

4. Поляков М. А. Конечные автоматы с небинарными элементами множеств [Електронний ресурс] / М. А. Поляков, И. А. Андриас // Системні технології. - 2019. - Вип. 2. - С. 85-94. - Режим доступу: http://nbuv.gov.ua/UJRN/st_2019_2_12

5. И.В. Вельбицкий Визуальное программирование графическими структурами [Заголовок з екрану]. – режим доступу http://emag.iis.ru/arc/infosoc/emag.nsf/BPA/e72abd849fe68a7dc32576eb0034c090

6. Цейтлин Г. Е. Алгебро-алгоритмические средства проектирования знаний предметных областей / Г. Е. Цейтлин, Л. М. Захария // Кибернетика и системный анализ. - 2009. - Т. 45, № 6. - С. 13-23. - Режим доступу: http://nbuv.gov.ua/UJRN/KSA_2009_45_6_3

References

1. Oskolkov A. P. Sety Petry – ynstrument dlia opysanyia y yssledovanyia dynamycheskykh system [Elektronnyi resurs] / A. P. Oskolkov // Systemy obrobky informatsii. - 2014. - Vyp. 6. - S. 149-151. - Rezhym dostupu: http://nbuv.gov.ua/UJRN/soi_2014_6_37

2. Shostak Y. V. Metod rasshyrenyia modely lohystycheskoi tsepy postavok, predstavlennoi v forme dvukhurovnevoi vlozhennoi sety Petry [Elektronnyi resurs] / Y. V. Shostak, Ya. Rakhymy // Radioelektronni i kompiuterni systemy. - 2019. - № 1. - S. 82–90. - Rezhym dostupu: http://nbuv.gov.ua/UJRN/recs_2019_1_11

3. Kryvyi S. L. Konechnye avtomaty v ynformatsyonnykh tekhnolohyiakh [Elektronnyi resurs] / S. L. Кгыvыi // Kybernetyka y systemnыi analyz. - 2011. - T. 47, № 5. - S. 3-20. - Rezhym dostupu: http://nbuv.gov.ua/UJRN/KSA_2011_47_5_3

4. Poliakov M. A. Konechnye avtomaty s nebynarnymy elementamy mnozhestv [Elektronnyi resurs] / M. A. Poliakov, Y. A. Andryas // Systemni tekhnolohii. - 2019. - Vyp. 2. - S. 85-94. - Rezhym dostupu: http://nbuv.gov.ua/UJRN/st_2019_2_12

5. Y.V. Velbytskyi Vyzualnoe prohrammyrovanye hrafycheskymy strukturamy [Zaholovok z ekranu]. – rezhym dostupu http://emag.iis.ru/arc/infosoc/emag.nsf/BPA/e72abd849fe68a7dc32576eb0034c090

6. Tseitlyn H. E. Alhebro-alhorytmycheskye sredstva proektyrovanyia znanyi predmetnыkh oblastei / H. E. Tseitlyn, L. M. Zakharyia // Kybernetyka y systemnыi analyz. - 2009. - T. 45, № 6. - S. 13-23. - Rezhym dostupu: http://nbuv.gov.ua/UJRN/KSA_2009_45_6_3