Всеукраїнський конкурс студентських наукових робіт

Спеціальність «Комп’ютерні науки»

СТУДЕНТСЬКА НАУКОВА РОБОТА

на тему:

«Дослідження моделей і методів пошуку повторно використовуваних функцій інформаційної системи»

Шифр: ***«Нейрон»***

2019 р.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД − база даних;

ВМ – візуальна модель;

ЖЦ – життєвий цикл;

ІС – інформаційна система;

ІТ – інформаційні технології;

НМ – нейронна мережа;

ООП – об’єктно-орієнтоване программування;

ПЗ – програмне забезпечення;

ПрО – предметна область;

ПЗ – програмне забезпечення;

УВ – управління вимогами;

ШІ – штучний інтелект;

ШНМ – штучна нейронна мережа;

AD – architecture description;

DFD − Data Flow Diagram;

UML – Unified Modelling Language.

ВСТУП

На даний час ринок IT-послуг досить нестабільний. Результати досліджень компанії Gartner показують, що в 2015-2016 роках витрати на IT-послуги відчутно скоротилися. Деяке зростання витрат на IT-послуги в 2017 році не усунуло невизначеність динаміки ринку [1]. Слід зазначити, що в даний час найбільшим сегментом ринку IT-послуг залишається сегмент програмного забезпечення (ПЗ) [1]. Тому однією з основних проблем, що вимагають уваги з боку споживачів і постачальників IT-послуг, є проблема скорочення витрат на виробництво IT-послуг.

Однією з найбільш важливих статей витрат IT-проектів створення програмних продуктів є стаття «Витрати на персонал». До даного виду витрат відносяться, зокрема, витрати на наймання персоналу для участі в IT-проект, на оплату праці персоналу IT-проекту, на навчання та підвищення кваліфікації персоналу IT-проекту. Слід також зазначити, що існуючі практики управління проектами рекомендують розділяти персонал проекту на групи постійних співробітників і співробітників, найманих для участі в конкретному проекті [2]. У той же час існуючі моделі зрілості IT-компаній припускають розвиток процесів розробки програмних продуктів та управління IT-проектом в напрямку забезпечення їх повторюваності і стандартизації [3, 4]. Такий підхід дозволяє висунути припущення про можливість заміни персоналу IT-проекту в ряді повторюваних процесів і робіт по розробці програмного продукту інтелектуальними інформаційними технологіями (IT) за умови економічної доцільності подібної заміни. Подібне рішення проблеми скорочення витрат на участь персоналу в розробці інформаційних систем (IC) і програмних продуктів різного призначення є актуальним з теоретичної та прикладної точок зору.

Таким чином, метою конкурсної роботи є дослідження моделей і методів пошуку повторно використовуваних функцій інформаційної системи.

Досягнення цієї мети дозволить скоротити витрати на участь персоналу у розробці інформаційних систем та програмних продуктів різного призначення.

Для досягнення поставленої мети необхідно вирішити такі задачі:

− провести аналіз існуючих підходів до повторного використання елементів ІС;

− провести аналіз існуючих інструментальних засобів макропроектування ІС;

− провести аналіз математичних моделей та методів рішення задачі повторного використання елементів ІС;

− обґрунтувати вибір нейрону для вирішення задачі повторного використання елементів інформаційної системи;

− розробити модель вирішуючої функції нейрону;

− розробити моделі блоку навчання нейрону;

− розробити алгоритм навчання нейрону та алгоритму розрахунку вирішуючої функції;

− розробити алгоритм рішення задачі пошуку повторного використовуваного елементу з використання нейрону ADALINE;

− провести апробацію отриманих результатів.

Науковим результатом є моделі пошуку повторно використовуваних функцій інформаційної системи, які дозволяють реалізувати вирішення цієї задачі у вигляді модифікованого нейрону mADALINE.

Під час проведення досліджень результати даної роботи були опубліковані в одній статті в журналі з переліку фахових видань України, який індексується у науковометричній базі даних Scopus [5], та у двох тезах доповідей на міжнародних наукових конференціях [6-7].

1 АНАЛІЗ ПРОБЛЕМИ ПОВТОРНОГО ВИКОРИСТАННЯ ФУНКЦІЙ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

На даний час створення ІС стандартизовано. Згідно стандарту ISO / IEC 15288-2005 система − це комбінація взаємодіючих елементів, організованих для досягнення однієї або декількох поставлених цілей [8]. Під елементом системи розуміємо представника сукупності елементів, що утворюють систему, при цьому елемент системи є окремою частиною системи, що дозволяє стверджувати про можливість створення системи з вже існуючих або повторно використовуваних елементів, за умови що ці елементи відповідають вимогам що висуваються до системи [8].

З сучасної точки зору ІС є системами, утвореними множиною взаємопов'язаних і економічно доцільних ІТ-послуг, своєчасне надання і виконання яких забезпечує ефективну і якісну діяльність керованого об'єкта і / або процесу. Підхід, на підставі якого сформовано дане подання ІС, тут і надалі будемо називати сервісним підходом. Даний підхід дозволяє встановити відповідність економічно і технічно обґрунтованих потреб зацікавлених осіб в автоматизації управління об'єктами і / або процесами безлічі конкретних ІТ-послуг і реалізують ці послуги ІТ-сервісів, що дозволяють задовольнити ці потреби [9].

Сервісний підхід до створення ІС як множини взаємозалежних елементів, здатних надавати зацікавленим особам послуги з виконання необхідних для досягнення головної мети діяльності ІС операцій над даними можна описати так: кожна проблема зацікавлених осіб може бути виражена множиною потреб різної природи, висловлюваних зацікавленими сторонами в різній формі. Кожна з цих потреб може бути задоволена однією або декількома ІТ-послугами або ж одним або декількома ІТ- сервісами, що реалізують відповідні ІТ-послуги. Тоді будь-яка ІС може бути представлена як система, що складається з множини ІТ-

послуг, а будь-яка IT-послуга може бути реалізована множиною IT-сервісів [10].

Подібне уявлення ІС дозволяє встановити єдину взаємопов'язану термінологічну базу для опису функціональної структури і забезпечує частини ІС.

В даний час одним з найкращих способів мінімізації витрат під час виконання IT-проектів вважається повторне використання елементів, розроблених в попередніх проектах, для задоволення вимог, висунутих до нового проекту. Даний спосіб вимагає рішення задачі вибору такої конфігурації ІС, яка в максимально можливій мірі могла б бути реалізована із застосуванням повторно використовуваних елементів.

На ранніх етапах життєвого циклу (ЖЦ) ІС під елементом будемо розуміти функції або окремі IT-послуги.

В умовах дефіциту фінансових ресурсів і високих ризиків створення і впровадження ІС і технологій особливого значення набувають рішення, що дозволяють з безлічі можливих функцій автоматизувати виконання тільки тих, які справді необхідні підприємству-замовнику.

При створенні проектів складних ІС, що складаються з багатьох елементів, кожен з яких може мати різновиди або версії, виникає проблема обліку їх зв'язків і функцій, створення уніфікованої структури і забезпечення розвитку всієї системи. Кожен процес характеризується певними завданнями і методами їх вирішення, вихідними даними, отриманими в ході виконання попереднього процесу, а також результатами. Управління конфігурацією дозволяє організувати, систематично враховувати і контролювати внесення змін до програмного забезпечення на всіх стадіях ЖЦ [6].

Рішення завдання вибору конфігурації ІС починається з визначення сервісу, який повинен забезпечуватися системою, і рівня сервісу, який може забезпечити дана конфігурація. Маючи набір цільових показників продуктивності кінцевого користувача і вартісних обмежень, необхідно спрогнозувати можливості певного набору елементів, які включаються в конфігурацію системи.

В даний час системи та ІТ, орієнтовані на автоматизацію робіт по формуванню, аналізу та управління вимог (УВ), досить поширені. Область застосування таких систем і ІТ охоплює широкий діапазон ІТ-проектів - від складних проектів з великою кількістю учасників до найпростіших проектів, які виконуються однією людиною або ж командою з кількох виконавців.

В цілому ж в [11] відзначається, що в даний час спостерігається поділ продуктів, які традиційно відносяться до систем УВ, на два різних напрямки. Продукти першого напряму працюють з вимогами апріорно заданих типів і не можуть бути розширені або адаптовані під вимоги конкретних користувачів. Продукти другого напрямку засновані на принципі відкритих моделей і дозволяють описувати не тільки вимоги до створюваної системи, а й інші артефакти, які використовуються для опису системи (наприклад, ризики, тести, помилки і т.п.).

Однак жодна з існуючих систем УВ не орієнтована на автоматизацію синтезу architecture description (AD) створюваної системи. Крім того, проблема прийняття рішень про ре-використання раніше реалізованих вимог в нових ІТ-проектах як і раніше далека в даних системах від ефективного вирішення. Вирішення цих проблем дозволить значно скоротити витрати часу і інших ресурсів за рахунок автоматизованого формування оптимальних (або раціональних) варіантів AD створюваної системи, в яких раніше реалізовані вимоги ре-використовуються в максимально можливою для створюваної системи ступеня [9].

Під час аналізу математичних моделей і методів повторного використання функцій ІС було виділено два основні підходи до вирішення задачі. Перший підхід полягає у використанні для пошуку і виділення повторно використання функцій опису цих функцій різного ступеня формалізованості. Другий підхід полягає у виділенні і повторному використанні знань про ці функції, при чому опису цих знань засновані на одній і комбінації декількох моделей подання знань.

Класичним прикладом першого підходу можна вважати використань положень доменного моделювання. Прикладом другого підходу є розглянуті в [9] моделі і методи формування використання уявлень функціональних вимог до ІС на рівні знань. В основі даних моделей лежить модифікована фреймова модель знань.

Фрейм є структурою даних для представлення стереотипної ситуації. У методології об'єктно-орієнтованого програмування (ООП) дане поняття відповідає класу [12].

Як правило, фреймові моделі ПрО представляються у вигляді мережі фреймів, що складається з вузлів і різних зв'язків між ними [12, 13]:

$$M =< FR, C, G >, \tag{1.1}$$

де $FR = \{fr_1, \ldots, fr_h\}$ – множина інформаційних одиниць (фреймів);

$C = \{C_1, C_2, \ldots, C_n\}$ – множина зв'язків та відношень між інформаційними одиницями (ієрархічні, посилальні и т.д.);

G – множина відображень, які задають зв'язки із заданої множини $\{C_1, C_2, \ldots, C_n\}$ між інформаційними одиницями, що входять у множину FR; $G_i \in G =< fr_{i1}, fr_{i2}, C_i >$ [14].

Фрейм $fr \in FR$ можна описати структурованою множиною, що має наступний вигляд[13, 15]:

$$fr = \{ n, [(ns_1, vs_1, ps_1), (ns_2, vs_2, ps_2), \ldots (ns_k, vs_k, ps_k)] \}, \tag{1.2}$$

де n – найменування фрейму;

(ns, vs, ps) – слот фрейму;

k – кількість слотів фрейму;

ns_i – найменування слоту, $i = \overline{1,k}$;

vs_i – значення слоту, $i=\overline{1,k}$;

ps_i – найменування приєднаної процедури, $i=\overline{1,k}$.

Як значення слоту «найменування приєднаної процедури» в фреймах використовується підпрограма процедурного типу. Приєднаним процедурам в ООП відповідають методи класів, в реляційних БД їм відповідають пов'язані з таблицями тригери, процедури і функції [16].

Однак фреймова модель в класичному вигляді, представленому виразом (1.2), не повною мірою відповідає особливостям представлення знань про ПрО, і особливо про створювану ІС і її елементи. Подібні особливості визначаються, головним чином, тими парадигмами проектування систем, які поміщаються Постачальником в основу переважної більшості проектних рішень ІС в цілому, окремих ІТ-послуг та ІТ-сервісів. В якості таких парадигм зазвичай вказуються структурний і об'єктно-орієнтований підходи. Дані підходи, в свою чергу, визначають необхідність модернізації фреймової моделі подання знань про ПрО, ІС, ІТ-послуги та ІТ-сервісах таким чином, щоб ці знання можна було згодом перетворити в паттерни і конкретні проектні рішення по видам забезпечень створюваної ІС.

Перш за все, пропонується розширити базове поняття «фрейм» шляхом виділення як окремого елемента фрейму сукупності всіх методів (приєднаних процедур) $Mt=\{mt_1,\ldots,mt_z\}$, зв'язаних з фреймом у цілому, а не з конкретними слотами.

Крім того, пропонується розширити базове поняття «фрейм» шляхом введення додаткової поняття «інтерфейс фрейму», відповідного поняттю «інтерфейс класу» в методології ООП [13, 16]. Слід зазначити, що в деяких мовах програмування підтримується можливість визначення властивостей в інтерфейсах класів. В інших мовах програмування властивості можуть бути оголошені шляхом завдання методів $GetPropertyName$ та $SetPropertyName$, що здійснюють доступ до полів класу. Інтерфейси, як і класи, можуть містити посилальні властивості і успадковуватися, утворюючи ієрархії [13, 17-19].

У реляційних БД немає поняття, повністю відповідного терміну «інтерфейс класу». Найбільш близьким до нього з точки зору участі в організації взаємодії між різними компонентами ІС є термін «подання».

Сказане вище дозволяє визначити термін «інтерфейс фрейму» як декларативне оголошення множини властивостей і методів без їх докладного опису (в тому числі, без докладного опису приєднаних процедур). Передбачається, що кожен інтерфейс фрейму описує окрему точку зору на фрейм або їх підмножину, причому дана точка зору може не сприймати ці фрейми або підмножину фреймів повністю.

Відповідно до запропонованого визначення, формалізований опис інтерфейсу фрейму if буде являти собою структуровану множину, що має такий вигляд [13, 16]:

$$if = < g, \{ns\_if_1, ..., ns\_if_n\}, \{nm_1, ..., nm_s\} >, \quad (1.3)$$

де g – глобально унікальний ідентифікатор;

$\{ns\_if_1, ..., ns\_if_n\}$ – множина декларативних оголошень слотів в інтерфейсі фрейму;

$\{nm_1, ..., nm_s\}$ – множина декларативних оголошень методів в інтерфейсі фрейму.

Слід зазначити, що будь-якій декларативно оголошеному найменуванню слоту інтерфейсу фрейму можна поставити у відповідність множину значень як окремого фрейму $fr \in Fr$, так і окремого слоту $ns_i \in fr$. Надалі фрейм, який бере участь у формуванні інтерфейсу if, будемо називати породжуючим фреймом.

Тоді формалізований опис поняття «фрейму» (1.2) с урахуванням запропонованих модифікацій прийме наступний вигляд [13, 16]:

$$fr = \{ n, [(ns_1, vs_1, ps_1), ... (ns_k, vs_k, ps_k)], \{ if_1, ..., if_n \}, \{ mt_1, ..., mt_z \} \}, \quad (1.4)$$

де $\{if_1,...,if_n\}$ – множина інтерфейсів, що використовуються фреймом fr (може бути порожнім).

Використання модифікованого формалізованого опису фрейму (1.4) дозволяє стверджувати, що мережа фреймів (1.1) буде включати не тільки знання про структури даних, що подаються у вигляді фреймів, а й знання про процеси, в ході яких буде здійснюватися взаємодія цих структур. Такі знання представляються в мережі у вигляді інтерфейсів і методів фреймів. Крім того, використання модифікованого опису фрейму дозволяє формалізувати не тільки опису знань про ПрО, інформаційне забезпечення і ПЗ створюваної ІС, а й опису їх взаємно-однозначних відображень [9].

Таким чином можна стверджувати, що основною особливістю рішення задачі повторного використання функцій ІС є орієнтація на представлення знань про ці функції одним з розглянутих вище способів.

Проведений аналіз дозволяє зробити висновок про перспективність досліджень щодо вирішення завдань класифікації, ідентифікації та пошуку повторно використовуваних елементів у ході виявлення та аналізу вимог до створюваних продуктів. Подібні дослідження дозволять значно підвищити точність оцінювання витрат і скоротити витрати часу на розробку ІС і програмних продуктів.

Виходячи зі сказаного вище, основна задача дослідження полягає в дослідженні моделей та методів, що дозволяють вирішити задачу повторно використовуваних функцій інформаційної системи. Для досягнення цієї мети необхідно вирішити такі задачі дослідження:

− аналіз проблеми повторного використання функцій інформаційної системи та постановка задачі дослідження;

− дослідження особливостей рішення задачі повторного використання елементів інформаційної системи з застосуванням нейрона ADALINE ;

− розробка алгоритму рішення задачі пошуку повторного використовуваного елемента ;

− практична реалізація, апробація результатів конкурсної роботи.

2 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ РІШЕННЯ ЗАДАЧІ ПОВТОРНОГО ВИКОРИСТАННЯ ЕЛЕМЕНТІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ЗАСТОСУВАННЯМ НЕЙРОНА ADALINE

На даний час проблема застосування інтелектуальних ІТ для автоматизації процесів розробки програмних продуктів різного призначення є однією з найбільш перспективних проблем наукового дослідження в ІТ-сфері. При цьому в якості основних інструментів ШІ, придатних для вирішення даної проблеми, розглядаються різні нейронні мережі (НМ).

Основною областю, в якій застосування інтелектуальних ІТ є найбільш виправданим, більшість дослідників вважають роботи по класифікації та ідентифікації повторно використовуваних елементів програмних продуктів.

В даний час існує досить велика кількість видів нейронів. Однак для перевірки принципової можливості їх застосування в задачі пошуку опису повторно використовуваної функції для реалізації функціонального вимоги, висунутої до ІС, необхідно використовувати максимально прості в реалізації нейрони.

Для вирішення задачі пошуку повторно використовуваної функції ІС, пропонується використовувати НМ, яка, на основі описів вхідних і вихідних документів,що надходять, для кожної окремої функції ІС, буде оцінювати можливість реалізувати цю функцію шляхом використання повторних функцій.

Було проведено аналіз основних НМ. Результати порівняльного аналізу нейронних мереж ADALINE, Хопфілда та Хеммінга наведено у таблиці 2.1.

На основі проведеного порівняльного аналізу основних існуючих видів нейронів, було виявлено, щоб застосування нейрона типу ADALINE слід вважати найбільш простою технологією, що дозволяє оцінити принципову

Таблиця 2.1 − Порівняльний аналіз нейронних мереж ADALINE, Хопфілда та Хеммінга

НМ	Кількість прошарків	Тип вхідних сигналів	Тип вихідних сигналів	Складність реалізації	Точність рішення
НМ ADALINE	1	Цілі числа	Аналоговий та бінарний	Самий простий	Найменша
НМ Хопфілда	1	Бінарні вектори	Цілі числа	Складний	Середня
НМ Хеммінга	2	Бінарні вектори	Цілі числа	Складний	Середня

можливість вирішення завдання формування варіанти конфігурацій за допомогою НМ [6]. Нейрони даного виду можуть використовуватися як в якості елементарних нейронів у складі НМ, так і самостійно в задачах розпізнавання образів, обробки сигналів, реалізації логічних функцій [20, 21].

Структурна схема нейрона ADALINE приведена на рисунку 2.1 [20, 21].

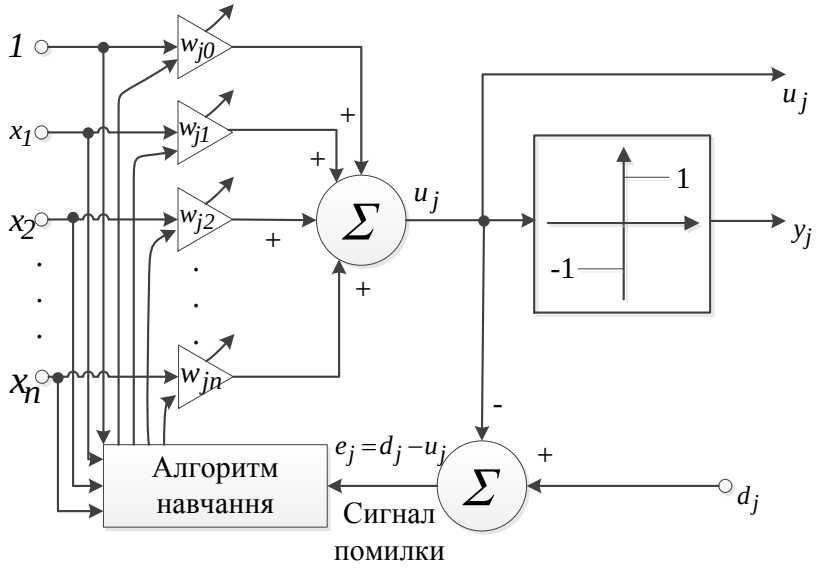

Рисунок 2.1 − Структурна схема нейрона ADALINE

Слід зазначити, що вихідними даними для НМ у ході вирішення задач класифікації, ідентифікації та пошуку повторно використовуваних елементів програмних продуктів є їх формальні описи. Такі описи можуть бути представлені наступним чином:

− спеціальна модель, заснована на кількісних метрики оцінювання повторного використання програмних елементів [22];

− уявлення елементу або сервісу у вигляді безлічі формальних описів їх функцій або інтерфейсів [23,24];

− формальний опис елемента на основі діаграм класів UML [25].

Останній з розглянутих способів особливо перспективний, оскільки дозволяє вирішувати завдання класифікації, ідентифікації та пошуку повторно використовуваних елементів вже в ході виявлення і аналізу вимог до створюваних продуктів. При цьому виникає додаткова задача формального опису вимоги до продукту або до ІС, реалізація якого можлива шляхом повторного використання готового елементу. В якості таких описів пропонується використовувати знання-орієнтовані описи вимог. В [26] для подання функціональних вимог на рівні знань запропоновано використовувати паттерни, засновані на мережах фреймів.

Модель подання *i*-ї функціональної вимоги (ФВ) до ІС tr_i^f на рівні знань, представляє собою множину кортежів вигляду[27]:

$$K_i^f = \{< d_n^{ij}, \{< d_{el\_fr}^{ij}, d_{el\_fr\_t}^{ij} >\} >, < d_g^{ij}, \{< d_{el\_if}^{ij}, d_{el\_if\_t}^{ij} >, \} >, < d_{fr\_rel\_n}^{ij}, \{< d_{el\_fr\_rel}^{ij}, d_{el\_fr\_rel\_t}^{ij} >\} >\}, \quad (2.1)$$

де d_n^{ij} – опис найменування фрейму;

$d_{el\_fr}^{ij}$ – опис елементу фрейму;

$d_{el\_fr\_t}^{ij}$ – опис типу елементу фрейму;

d_g^{ij} – опис найменування інтерфейсу;

$d_{el\_fi}^{ij}$ – опис елементу інтерфейсу;

$d_{el\_fi\_t}^{ij}$ – опис типу елементу інтерфейсу;

$d_{fr\_rel\_n}^{ij}$ – опис назви зв'язку між інтерфейсами й/або фреймами;

$d_{el\_fr\_rel}^{ij}$ – опис елементу зв'язку;

$d_{el\_fr\_rel\_t}^{ij}$ – опис типу елементу зв'язку.

При цьому між елементами цього зображення існують наступні відношення належності [27]:

а) відношення належності елементів фрейму конкретному фрейму, що має вигляд:

$$F^{d_n^{ij}}_{\{<d_{el\_fr}^{ij},d_{el\_fr\_t}^{ij}>\}}:\begin{cases} D_n^i \to 2^{\{<d_{el\_fr}^{ij},d_{el\_fr\_t}^{ij}>\}} \\ d_n^{ij} \to \{<d_{el\_fr}^{ij},d_{el\_fr\_t}^{ij}>\} \end{cases},$$

де D_n^i – множина описів найменувань фреймів, що характеризують концепти ПрО з погляду окремого учасника автоматизованого процесу st_k;

$2^{\{<d_{el\_fr}^{ij},d_{el\_fr\_t}^{ij}>\}}$ – булеан (множина всіх підмножин) підмножини $\{<d_{el\_fr}^{ij},d_{el\_fr\_t}^{ij}>\}$, що описує можливий вміст фреймів з множиною описів найменувань D_n^i;

$\{<d_{el\_fr}^{ij},d_{el\_fr\_t}^{ij}>\}$ – підмножина описів елементів і їх типів фрейму з найменуванням d_n^{ij};

б) відношення належності елементів інтерфейсу конкретному інтерфейсу, що має вигляд:

$$F^{d_g^{ij}}_{\{<d_{el\_fi}^{ij},d_{el\_fi\_t}^{ij}>\}}:\begin{cases} D_g^i \to 2^{\{<d_{el\_fi}^{ij},d_{el\_fi\_t}^{ij}>\}} \\ d_g^{ij} \to \{<d_{el\_fi}^{ij},d_{el\_if\_t}^{ij}>\} \end{cases},$$

де D_g^i – множина описів найменувань інтерфейсів, що характеризують концепти ПрО з погляду окремого УАП st_k;

$2^{\{<d_{el\_fi}^{ij},d_{el\_fi\_t}^{ij}>\}}$ – булеан підмножини $\{<d_{el\_fi}^{ij},d_{el\_fi\_t}^{ij}>\}$, що описує можливий вміст інтерфейсів з множиною описів D_g^i;

$\{<d_{el\_fi}^{ij},d_{el\_fi\_t}^{ij}>\}$ – підмножина описів елементів і їх типів інтерфейсу з найменуванням d_g^{ij}.

в) відношення належності елементів зв'язку конкретному зв'язку, що має вигляд:

$$F^{d_{fr\_rel\_n}^{ij}}_{\{<d_{el\_fr\_rel}^{ij},d_{el\_fr\_rel\_t}^{ij}>\}}:$$
$$:\begin{cases} D_{fr\_rel\_n}^i \to 2^{\{<d_{el\_fr\_rel}^{ij},d_{el\_fr\_rel\_t}^{ij}>\}}; \\ d_{fr\_rel\_n}^{ij} \to \{<d_{el\_fr\_rel}^{ij},d_{el\_fr\_rel\_t}^{ij}>\} \end{cases},$$

де $D_{fr\_rel\_n}^i$ – множина описів найменувань зв'язків між інтерфейсами й/або фреймами, що характеризують концепти ПрО з погляду окремого УАП st_k;

$2^{\{<d_{el\_fr\_rel}^{ij},d_{el\_fr\_rel\_t}^{ij}>\}}$ – булеан підмножини $<d_{el\_fr\_rel}^{ij},d_{el\_fr\_rel\_t}^{ij}>$, що описує можливий вміст описів зв'язків із множиною описів найменувань $D_{fr\_rel\_n}^i$;

$< d^{ij}_{el\_fr\_rel}, d^{ij}_{el\_fr\_rel\_t} >$ – підмножина описів елементів і їх типів зв'язку з найменуванням $d^{ij}_{fr\_rel\_n}$.

Ця модель дозволяє описати подання функціональних вимог до ІС на рівні знань. Її використання дозволяє спростити автоматизацію операцій над поданнями функціональних вимог до ІС на рівні знань за рахунок уніфікації їх формального опису [9].

Щоб застосувати ADALINE для вирішення завдання пошуку опису повторно використовуваної функції для реалізації функціонального вимоги, висунутого до ІС, необхідно визначити:

а) формальний опис вхідних сигналів x_j;

б) формальний опис навчального сигналу d_j;

в) формальний опис вагових коефіцієнтів w_j;

г) формальний опис вихідного аналогового сигналу u_j;

д) модель адаптивного лінійного асоціатор;

е) конкретний вид нелінійної активаційної функції y_j;

ж) конкретний вид алгоритму навчання.

В загальному випадку задача пошуку опису повторно використовуваної функції для реалізації функціональної вимоги, що ви висунута до ІС, потребує порівняння формальних описів повторно використовуваних функцій та формального опису функціональної вимоги, що висунута до ІС Споживачем ІТ-послуг. Тоді у якості вхідних сигналів x_j пропонується розглядати подання j-ої функціональної вимоги до ІС, що була реалізована раніше, на рівні знань $K_j^{f_{lib}}$. У якості навчального сигналу d_j пропонується розглядати подання i-ої функціональної вимоги до ІС на рівні знань $K_i^{tr_U}$ Споживача. У якості опису вихідного аналогового сигналу u_j пропонується розглядати загальносистемне подання j-ої функціональної вимоги до ІС на рівні знань $K_i^{f_{IS}}$. Формальний опис $K_j^{f_{lib}}$, $K_i^{tr_U}$ и $K_i^{f_{IS}}$ у вигляді мереж фреймів приведено у [28].

Слід зазначити, що уявлення вимог на рівні знань $K_j^{f_{lib}}$, $K_i^{tr_U}$ і $K_i^{f_{IS}}$ є сукупностями числових і символьних даних. Паттерни проектування цих подань розглянуті в [26]. Однак нейрон ADALINE може обробляти тільки числові дані. Перетворення подань $K_j^{f_{lib}}$ и $K_i^{tr_U}$ у вектора числових сигналів потребує доволі складних алгоритмів. Тому пропонується модифікувати ADALINE так, щоб він міг обробляти дані подання на основі їх параметрів, що обчислюються в ході роботи адаптивного лінійного асоціатору. Надалі модифікований нейрон ADALINE будемо позначати як mADALINE.

Метою рішення задачі пошуку опису повторно використовуваної функції є знаходження такої функції, повторне використання якої вимагало б мінімальних витрат на адаптацію знайденої функції до особливостей висунутої функціонального вимоги. Дану мету можна уявити як пошук такого опису повторно використовуваної функції, яке в максимально можливій мірі дублює опис висунутого до ІС функціонального вимоги. Тому для розробки формального опису адаптивного лінійного асоціатору пропонується використовувати опис задачі виявлення дубльованих функціональних вимог, розглянуте в [28, 29].

У загальному випадку задача виявлення дубльованих вимог може бути формально описана наступним чином. Нехай є мережа фреймів $Arch_{base}$, що складається з множини подання $K_i^{f_{IS}}$. Кожне подання $K_i^{f_{IS}}$ є набором об’єктів ob (фреймів, інтерфейсів та зв’язків), опис якого розглянуто у [27]. Множина шуканих описів ІТ-послуг IT_{acm} є розбиття мережі $Arch_{base}$, таке, що

$$\{IT_{acm_1},\ldots,IT_{acm_k}\} = Arch_{base}$$

та

$$IT_{acm_i} \neq \varnothing \wedge IT_{acm_i} \cap IT_{acm_j} = \varnothing \text{ , для } 1 \leq i, j \leq k$$

де k – кількість ІТ-послуг ІС, що створюється. Кожна ІТ-послуга IT_{acm_j} має наступні характеристики:

а) $D(IT_{acm_j})$ – множина унікальних об'єктів;

б) $Occ(ob, IT_{acm_j})$ – кількість входжень об'єкта ob в опис ІТ-послуги IT_{acm_j};

в) $S(IT_{acm_j}) = \sum_{ob \in D} (IT_{acm_j}) Occ(ob, IT_{acm_j})$;

г) $W(IT_{acm_j}) = | D(IT_{acm_j}) |$;

д) $H(IT_{acm_j}) = S(IT_{acm_j}) / W(IT_{acm_j})$;

Функція вартості, що дозволяє оцінити ступінь дублювання подання $K_i^{f_{IS}}$, має у загальному випадку наступний вигляд [28, 29]:

$$\Pr ofit(IT_{acm}, r) = \frac{\sum_{j=1}^{k} \frac{S(IT_{acm_j})}{W(IT_{acm_j})^r} \times | IT_{acm_j} |}{\sum_{j=1}^{k} | IT_{acm_j} |} \qquad (2.2)$$

де $| IT_{acm_j} |$ – кількість загальносистемних подань функціональних вимог на рівні знань $K_i^{f_{IS}}$, що описують ІТ-послугу IT_{acm_j}; r – коефіцієнт відштовхування, $r \geq 1$.

Тоді постановку задачі синтезу варіантів описів архітектури створюваної ІС можна сформулювати наступним чином: для заданих $D(Arch_{base})$ та r знайти розбиття: IT_{acm}:$Profit(IT_{acm}, r) \in [Profit_{\max} - \varepsilon; Profit_{\max}]$, де $Profit_{max}$ – максимальне значення функції (2.2), ε – величина припустимої похибки. У випадку, якщо величину ε не вдається визначити за допомогою експертних оцінок, рекомендується вважати $\varepsilon = 0{,}1 \cdot Profit_{\max}$ [28, 29].

На відміну від завдання виявлення дубльованих вимог, завдання пошуку опису повторно використовуваної функції для реалізації функціонального вимоги, висунутого до ІС, вимагає попарного порівняння подання $K_i^{tr_U}$ з кожним з наявних у розпорядженні Постачальника ІТ-послуг поданням $K_j^{f_{lib}}$. Для опису такого порівняння перетворимо функцію вартості (2.2) до наступного вигляду:

$$\Pr ofit(IT_{acm}, r) = \frac{1}{\sum_{j=1}^{k} | IT_{acm_j} |} \times \sum_{j=1}^{k} \left(| IT_{acm_j} | \times \frac{S(IT_{acm_j})}{W(IT_{acm_j})^r} \right) \qquad (2.3)$$

З урахуванням сказаного вище параметри вираження (2.3) візьмуть наступні значення:

а) кількість досліджуваних ІТ-послуг (функцій створюваної ІС) *k = 1*;

б) $IT_{acm_j} = (K_i^{tr_U}, K_j^{f_{lib}})$, оскільки ми описуємо одну функцію двома порівнюваними між собою вимогами, отже $| IT_{acm_j} | = 2$;

в) $r=1$ (Кожна висунута до ІС функціональна вимога реалізується як самостійний модуль створюваної ІС і не передбачає подальшої декомпозиції в ході проектування ІС і розробки інформаційного і програмного забезпечень ІС).

Тоді функція (2.2) для опису адаптивного лінійного асоціатору mADALINE прийме наступний вигляд:

$$u_j = \Pr ofit(IT_{acm_j}) = \frac{S(IT_{acm_j})}{W(IT_{acm_j})} \to 2 \qquad (2.4)$$

Значення функції (2.4) буде дорівнювати 2 у тому випадку, якщо опис повторно використовуваної функції $K_j^{f_{lib}}$ в точності відповідає опису

функціональної вимоги до ІС $K_i^{tr_U}$, що Споживачем. Тому d_j=2 для будь-якої можливої пари $K_i^{tr_U}$ и $K_j^{f_{lib}}$.

Для модифікації нейрона ADALINE найбільш сильно слід змінити формальне опис вагових коефіцієнтів w_j. Для пропонованої модифікації нейрона mADALINE w_j буде являти собою не вектор, а єдиний коефіцієнт, формальний опис якого матиме такий вигляд:

$$w_j = K_j^{f_{lib}} \cup K_i^{tr_U} \tag{2.5}$$

Тоді вираз (2.4) можна записати наступним чином:

$$u_j = \Pr ofit(w_j) = \frac{S(w_j)}{W(w_j)} \to 2 \tag{2.6}$$

Однак такий формальний опис вагового коефіцієнта має цінність тільки під час вирішення задачі пошуку опису повторно використовуваної функції для реалізації функціонального вимоги, висунутої до ІС. Для опису результатів вирішення даної задачі, як показано в [27], доцільніше використовувати загальносистемне подання функціональної вимоги до ІС на рівні знань. Дане подання може бути сформоване на основі результату обчислення w_j наступним чином:

$$K_j^{f_{IS}} = K_j^{f_{lib}} \cup (w_j \setminus K_j^{f_{lib}}) \tag{2.7}$$

Таке загальносистемне подання $K_j^{f_{IS}}$ в подальшому може використовуватися як основа для формування специфікації на доопрацювання

повторно використовуваної функції з урахуванням особливостей реалізованої функціональної вимоги, висунутої до ІС Споживачем ІТ-послуг.

Рішення задачі пошуку опису повторно використовуваної функції для реалізації функціональної вимоги, висунутої до ІС, в загальному випадку зводиться до вибору однієї з наступних альтернатив:

а) визнання відсутності повторно використовуваних функцій, подібних опису функціональної вимоги до ІС $K_i^{tr_U}$ висунутої Споживачем;

б) знаходження опису однієї повторно використовуваної функції $K_j^{f_{lib}}$, який максимально подібний опису функціональної вимоги до ІС $K_i^{tr_U}$ висунутої Споживачем;

в) знаходження опису двох та більше повторно використовуваних функцій, які максимально подібні опису функціональної вимоги до ІС $K_i^{tr_U}$ висунутої Споживачем.

Тоді опис нелінійної активаційної функції mADALINE y_j можна виконати на основі одиничної функції Хевісайду. Для даної задачі y_j матиме такий вигляд:

$$y_j = \begin{cases} 0, \Pr ofit(w_j) < 2 - \varepsilon(k); \\ 1/2, \Pr ofit(w_j) = 2 - \varepsilon(k); \\ 1, \Pr ofit(w_j) > 2 - \varepsilon(k); \end{cases} \quad (2.8)$$

де $\varepsilon(k)$ – величина припустимої похибки на k-ом кроці навчання [5].

Значення y_j приймає значення 0, якщо за результатами розрахунку функції $\Pr ofit(w_j)$ ми не знайшли жодного опису повторно використовуваного рішення, яке відповідало б опису функціональної вимоги з заданим рівнем точності.

Значення y_j приймає значення 1/2, якщо ми знайшли опис повторно використовуваного рішення, яке відповідало опису функціональної вимоги з мінімальним заданим рівнем точності.

Значення y_j приймає значення 1, якщо ми знайшли за результатами розрахунку функції $\Pr ofit(w_j)$ опис повторно використовуваного рішення, яке відповідає опису функціональної вимоги в максимально можливій мірі заданого рівня точності.

Результати модифікації моделі mADALINE, представлені виразами (2.5) − (2.8), вимагають зміни і структурної схеми даного нейрона. Дана зміна визначить в подальшому основні особливості реалізації mADALINE у вигляді програмного модуля інтелектуальної IT створення або модифікації IC різного призначення.

В ході модифікації структурної схеми mADALINE необхідно враховувати такі особливості:

а) бібліотека уявлень раніше реалізованих функціональних вимог до IC на рівні знань $K_j^{f_{lib}}$ представляє собою репозиторій, у якому зберігається кінцеве число цих подань;

б) подання *j*-ї функціональної вимоги до IC на рівні знань $K_i^{tr_U}$ Споживача поступає в mADALINE ззовні;

в) значення u_j, що розраховане по формулі (2.6) і значення $K_j^{f_{IS}}$ розраховане по формулі (2.7), слід передавати зовнішньому користувачеві тільки в разі $y_j>0$.

З урахуванням цих умов структурна схема нейрона mADALINE матиме вигляд, показаний на рисунку 2.2 [5].

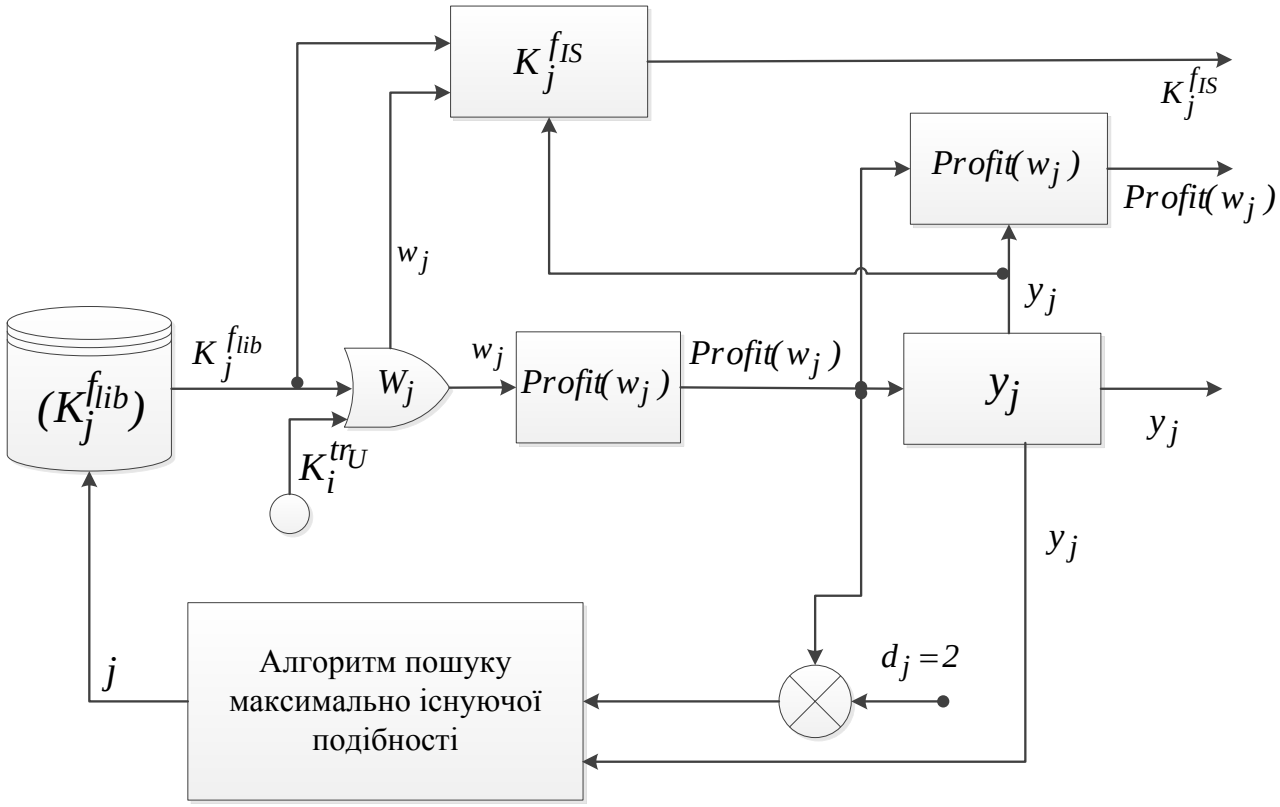

Рисунок 2.2 − Структурна схема модифікованого нейрону mADALINE

Опис нелінійної активаційної функції mADALINE y_j, вимагає звернути особливу увагу на вигляд алгоритму навчання. В результаті виконання задачі пошуку опису повторно використовуваної функції обробляються не тільки числові, а й символьні дані. Тому звичайні алгоритми (наприклад, нормалізований алгоритм методу найменших квадратів або алгоритм Уїдроу-Хоффа) не підходять для навчання mADALINE.

В якості найпростішого алгоритму навчання mADALINE в ході вирішення даної задачі пропонується алгоритм пошуку максимально існуючої подібності, що складається з наступних кроків.

Крок 0. Задати кількість ітерацій $k=0$ та максимально допустиме значення похибки $\varepsilon(k)=0{,}1$.

Крок 1. Прийняти $j=1$.

Крок 2. Якщо $\varepsilon(k)>2-Profit(w_j, k)$, то прийняти $\varepsilon(k)=2-Profit(w_j, k)$ та $k=k+1$.

Крок 3. Прийняти $j=j+1$. Якщо $j \le n$, то перейти до Кроку 2.

Крок 4. Якщо $y_j=0$, то прийняти $\varepsilon(k)=\varepsilon(k)+0{,}1$, в іншому випадку завершити роботу алгоритму.

Крок 5. Якщо $\varepsilon(k) \le 1$, то перейти до Кроку 1. В іншому випадку завершити роботу алгоритму.

В ході виконання даного алгоритму в першу чергу буде проводитися пошук доступних для повторного використання елементів, які збігаються з поданням функціональної вимоги на рівні знань більш ніж на 90%. Якщо таких компонентів не знайдено, ми знижуємо ступінь збігу до 80% і проводимо повторний пошук до тих пір, поки або не знайдемо елемент, що співпадає і описом сформульованої функціональної вимоги, або не прийдемо до висновку, що такі елементи в нашій базі відсутні.

В якості можливих способів удосконалення запропонованого алгоритму навчання можна вказати наступне:

а) визначення за результатами експлуатації нижньої межі допустимої ступеня збігу подання вимоги і опису повторно використовуваного елементу.

б) визначення найбільш ефективного кроку зміни значення допустимої похибки ε.

в) модифікація алгоритму з метою виділення і декількох компонентів, чиї описи збігаються з поданням вимоги, одного компонента, чиє опис збігаємося з поданням вимоги в максимальному ступені.

Розробка алгоритму рішенні задачі слід проводити з врахуванням можливості перенесення результатів розробки для наступної реалізації на різні програмні платформи. Тому необхідно дотримуватися вимоги платформо-незалежності розроблюваного алгоритму.

Слід також зазначити, що задача пошуку повторно використовуваних функцій ІС може вирішуватися не тільки під час формування та аналізу ФВ до ІС, а й під час виконання інших процесів ЖЦ ІС. Тому бажано розробити алгоритм розв’язання задачі пошуку повторно використовуваних функцій ІС

таким чином, щоб витрати на його модифікацію на вимоги інших процесів ЖЦ ІС були мінімальними.

Оскільки задача, яка вирішується в даній роботі має характер інноваційного дослідження, основну увагу слід приділяти пошуку найпростіших шляхів її вирішення, щоб потів в інших роботах обґрунтувати оптимальні алгоритми вирішення задачі пошуку повторно використовуваних функцій ІС. Тому розроблюваний алгоритм вирішуючої задачі має бути максимально простим для наочної демонстрації принципіальної можливості вирішення задачі пошуку повторно використовуваних функцій ІС.

Виходячи з цих умов, в якості найпростішого алгоритму розв'язання задачі пошуку повторно використовуваного елемента з використанням нейрона ADALINE пропонується алгоритм, що складається з наступних кроків.

Крок 0. Заповнення бази описів повторно використовуваних функцій.

Крок 1. Формування подання ФВ до ІС на рівні знань.

Крок 2. Формування опису ІТ послуги $w_{j.}$

Крок *3.* Розрахунок значення функції *Profit*(w_j).

Крок *4.* Розрахунок значення $y_{j.}$

Крок *5.* Якщо $y_j>0$, то зберегти y_j и j : $y_i \rightarrow \hat{y}_i$.

Крок *6.* Виконання алгоритму пошуку максимально існуючої подібності.

Крок *7.* Порівняння всіх збережених у.

Крок *8.* Якщо $\forall \hat{y}_j$ =1/2, то сформувати масив значень *Profit*($w_{j)}$ и $K_j^{f_{IS}}$. В іншому випадку перейти до Кроку 9.

Крок *9.* Видача масиву значень y_j ,*Profit*($w_{j)}$, $K_j^{f_{IS}}$. Завершення роботи нейрона.

У виконанні алгоритму розв'язання задачі пошуку повторно використовуваного елемента з використанням нейрона ADALINE приймають участь аналітик (Кроки 0-1), база описів повторно використовуваних функцій (Кроки 0-9) та нейрон ADALINE (Кроки 3-9)

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ, АПРОБАЦІЯ РЕЗУЛЬТАТІВ КОНКУРСНОЇ РОБОТИ

Задача пошуку повторно використовуваних функцій ІС є розширенням ІТ-послуги «Порівняння сформованих ієрархій термінів ПрО ІС з розробленими раніше ієрархіями термінів ПрО» [13].

Логічна схема даних задачі пошуку повторно використовуваних функцій інформаційної системи, представлена у вигляді ER-діаграми,яка наведена на рисунку 3.1.

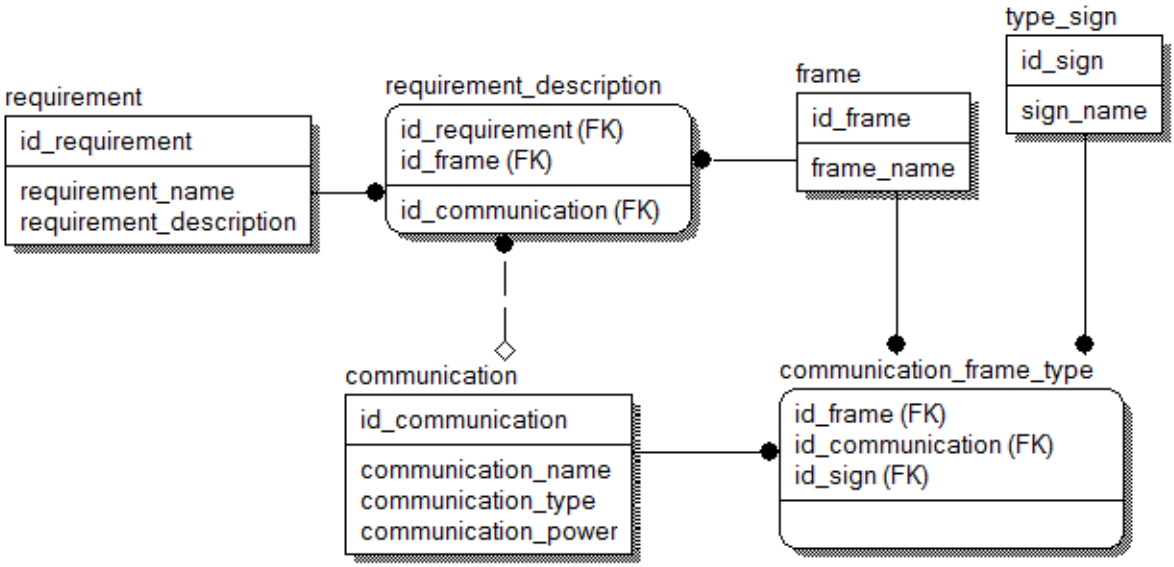

Рисунок 3.1 – Логічна схема даних задачі пошуку повторно використовуваних функцій інформаційної системи

Для кращого розуміння взаємодії між процесами та даними була побудована діаграма потоків даних (рисунок 3.2), яка базується на методології Data Flow Diagram (DFD).

Опис ходу та результатів апробації теоретичних результатів конкурсної роботи представлені у додатку А.

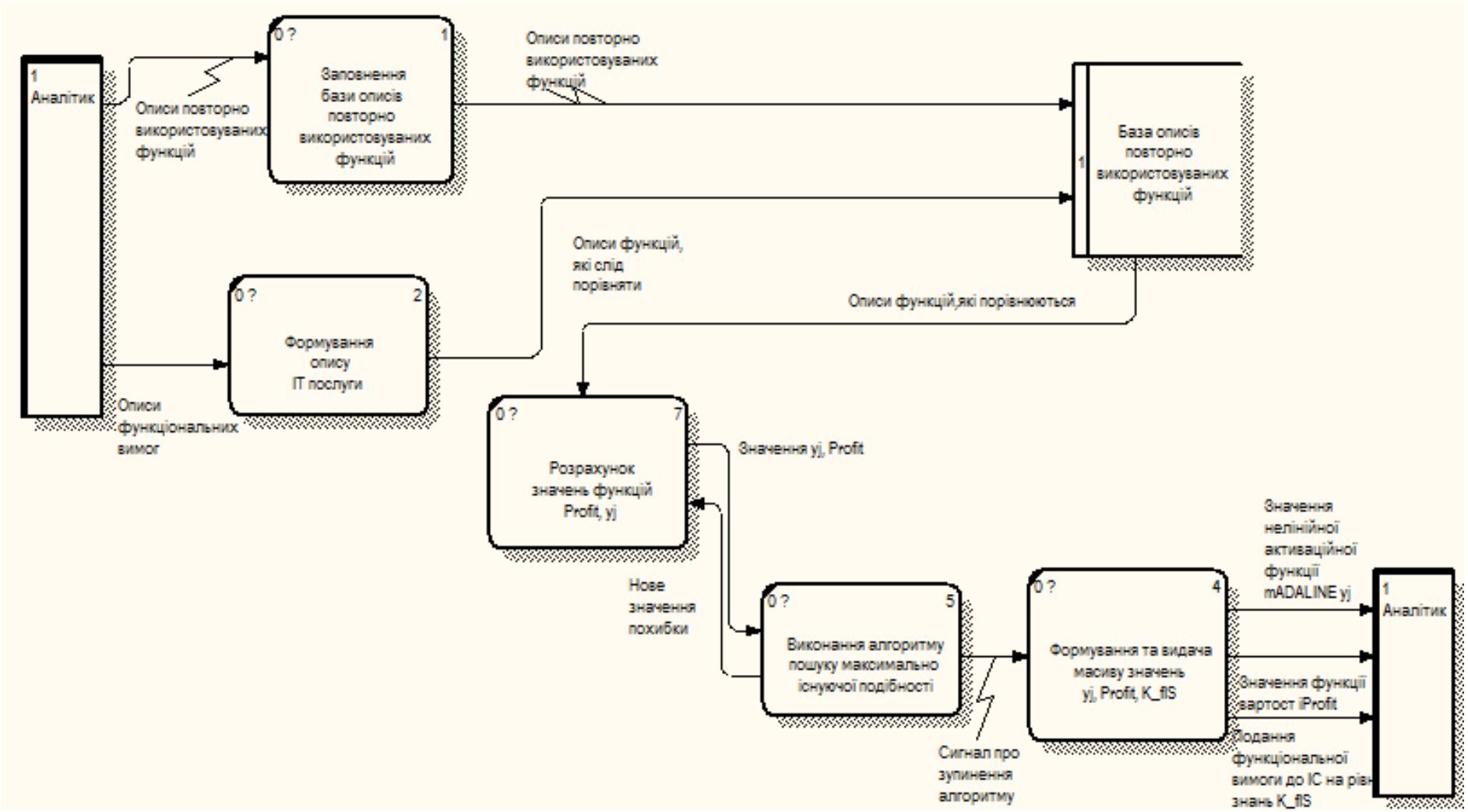

Рисунок 3.2 − Схема функціональної структури задачі пошуку повторно використовуваних функцій інформаційної системи (діаграма декомпозиції першого рівня)

ВИСНОВКИ

В конкурсній роботі було поставлено та вирішено актуальну задачу дослідження пошуку повторно використовуваних функцій ІС. Під час вирішення цієї задачі були отримані такі основні результати.

В результаті аналізу проблеми повторного використання функцій ІС, був зроблений висновок про перспективність досліджень щодо вирішення завдань класифікації, ідентифікації та пошуку повторно використовуваних елементів у ході виявлення та аналізу вимог до створюваних продуктів. В результаті проведеного аналізу існуючих підходів до повторного використання елементів інформаційної системи було виявлено, що для рішення задачі нам потрібно працювати не з самими елементами, тобто функціями ІС, а з їх формалізованими описами. Був проведен огляд існуючих інструментальних засобів макропроектування інформаційних систем.В ході аналізу математичних моделей і методів повторного використання функцій ІС було виділено два основні підходи до вирішення задачі.

Було проведено аналіз існуючих найпростіших НМ та обрана НМ ADALINE для вирішення задачі пошуку повторно використовуваних функцій ІС при умові модифікації структури та вирішуючої функції нейрону. Була розроблена модель вирішуючої функції нейрону, яка працює з представленням функціональної вимоги на рівні знань. У результаті модифікації моделі mADALINE, була змінена структурна схема нейрону ADALINE. Було запропоновано використовувати алгоритм пошуку максимально існуючої подібності у якості найпростішого алгоритму навчання mADALINE. Було розроблено найпростіший алгоритм розв'язання задачі пошуку повторно використовуваного елемента з використанням нейрона ADALINE для наочної демонстрації принципіальної можливості вирішення задачі пошуку повторно використовуваних функцій ІС. Була проведена апробація отриманих результатів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ИТ (мировой рынок) // TAdviser. Государство. Бизнес. ИТ. URL: http://www.tadviser.ru/index.php/Статья:ИТ_(мировой_рынок) (дата звернення 04.10.2018 р.)

2. Руководство к своду знаний по управлению проектами (Руководство PMBOK): пятое издание. Newton Square: Project Management Institute, Inc., 2013. − 586 с.

3. Бирюков А. Пять ступеней к совершенству // Директор информационной службы: электрон. версия журн. 2011. № 04. URL: http://www.osp.ru/cio/2011/04/13008116/ (дата звернення 04.10.2018 р.)

4. Терехов А. Современные модели качества программного обеспечения // Сайт компании «Interface». URL: http://www.interface.ru/fset.asp?Url=/misc/qs.htm (дата звернення 04.10.2018 р.).

5. Saif Q. Muhamed, Mohammed Q . Mohammed, Evlanov M., Kliuchko G. The Adaline neuron modification for solving the problem on searching for the reusable functions of the information system / Eastern-European Journal of Enterprise Technologies. 2018. Vol. 3. No 2 (93). − P. 25-32. (Наукометрична БД Scopus).

6. Ключко Г.Г. Исследование моделей и методов определения варианта конфигурации информационной системы / Г.Г. Ключко // 22-й Міжнародний молодіжний форум «Радіоелектроніка і молодь в XXI столітті». Зб. матеріалів форуму. Т. 6. – Харків: ХНУРЕ, 2018. – С. 133-134

7. Yevlanov M.V. Solving the problem on searching for the reusable functions of the information system / M.V. Yevlanov, Saif Q. Muhamed, Mohammed Q. Mohammed, Jane J. Eshaq, H.H. Kliuchko // I Міжнародна науково-практична конференція «Інформаційні системи та технології в медицині» (ISM-2018). Збірник наукових праць. ХНУРЕ. – Харків: «Друкарня Мадрид», 2018. – С. 200-202.

8. ГОСТ Р ИСО/МЭК 15288-2005. Информационная технология. Системная инженерия. Процессы жизненного цикла систем // Сайт АО «Кодекс». URL: http://docs.cntd.ru/document/gost-r-iso-mek-15288-2005 (дата звернення 15.09.2018 р.).

9. Евланов М.В. Модели, методы и информационная технология разработки архитектуры сложных информационных систем на основе функциональных тренований: дис. ... д-ра. техн. наук : 05.13.06 / З Харьковский национальный университет радиоэлектроники. Харків, 2017. − 429 с.

10. Евланов М.В. Онтологическая модель архитектуры информационной системы на основе сервисного подхода / М.В. Евланов // Радіоелектроніка, інформатика, управління. – 2013. - № 2. – С. 130-135.

11. Мадорская, Ю.М. Системы управления требованиями: что и зачем? / Ю.М. Мадорская // Сайт «ReqCenter.pro». URL: https://reqcenterpro (дата звернення 17.10.2018 р.).

12. Минский, М. Фреймы для представления знаний / М. Минский. –М.: Энергия, 1979. –152 с.

13. Левыкин В.М. Паттерны проектирования требований к информационной системе: моделирование и применение / В.М. Левыкин, М.В. Евланов, М.А. Керносов: монография. – Харьков: ООО «Компанія СМІТ», 2014. – 320 с.

14. Искусственный интеллект: в 3-х кн. Кн. 2. Модели и методы: Справочник / Под ред. Д.А. Поспелова – М.: Радио и связь, 1990. – 304 с.

15. Гаврилов, А.В. Системы искусственного интеллекта / А.В. Гаврилов. – Новосибирск: НГТУ, 2004. – 59 с.

16. Левыкин, В. М. Исследование и разработка фреймовой модели структуры документа / В. М. Левыкин, М. А. Керносов // Нові технології. – 2008. – № 1 (19). – С. 149–154.

17. Maciaszek, L.A. Requirements Analysis and System Design / L.A. Maciaszek: 2d ed. – Reading: Addison Wesley, Harlow England, 2005. – 504 p.

18. Deitel, H.M. C++ How to Program / H.M. Deitel, P.J. Deitel: 5th ed. – Pearson: Prentice Hall, Inc, 2005 – 1536 p.

19. Фленов, М.Е. Библия Delphi / М.Е. Фленов. – СПб.: БХВ-Петербург, 2004. – 880 с.

20. Руденко О.Г., Бодянский Е.В. Основы теории искусственных нейронных сетей / ТЕЛЕТЕХ. Харьков, 2002. − 317 с.

21. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы / Горячая линия-Телеком. М., 2004. − 452 с.

22. Singh C., Pratap A., Singhal A. Estimation of software reusability for component based system using soft computing techniques // Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit 6 November 2014. 2014. − P. 788-794. doi: 10.1109/CONFLUENCE.2014.6949307

23. Automated architectural component classification using concept lattices / Aboud N.A. etc. // Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, WICSA/ECSA. 2009. − P. 21-340. doi: 10.1109/WICSA.2009.5290788

24. Hwang Y. Facilitating service matchmaking based on semantic similarity // Information (Japan). 2015. V. 18. I. 11. − P. 4443-4458.

25. Panyangam B., Kiewkanya M. Software size estimation in design phase based on MLP neural network // Recent Advances in Information and Communication Technology. Proceedings of the 13th International Conference on Computing and Information Technology (IC2IT). 2017. − P. 82-91. doi: 10.1007/978-3-319-60663-7_8

26. Levykin V., Ievlanov M., Neumyvakina O. Developing the models of patterns in the design of requirements to an information system at the knowledge level // Восточно-Европейский журнал передовых технологий. 2017. № 5/2 (89). − С. 19-26. doi: 10.15587/1729-4061.2017.110586

27. Евланов, М.В. Разработка методов формирования представления сформулированного требования к информационной системе на уровне знаний / М.В. Евланов // Восточно-европейский журнал передовых технологий. – 2015. - № 4. – С. 4-11.

28. Евланов М.В. Разработка модели и метода выбора описания рациональной архитектуры информационной системы // Восточно-европейский журнал передовых технологий. 2016. № 1/2(79). − С. 4-12. doi: 10.15587/1729-4061.2016.60583

29. Євланов М.В., Васильцова Н.В., Панфьорова І.Ю. Моделі і методи синтезу опису раціональної архітектури інформаційної системи // Вісник наукового університету «Львівська політехніка». Серія «Інформаційні системи та мережі». 2015. № 829. − С. 135-152.

ДОДАТОК А

ОПИС ХОДУ ТА РЕЗУЛЬТАТІВ АПРОБАЦІЇ ТЕОРЕТИЧНИХ РЕЗУЛЬТАТІВ КОНКУРСНОЇ РОБОТИ

Розглянемо приклад використання запропонованого нейрону mADALINE.

Він полягає в експериментальному підтвердженні принципової можливості використання нейрону для вирішення задачі пошуку повторно використовуваних функцій ІС.

Для цього у якості повторно використовуваної функції пропонується узяти функцію «Облік гравців». У якості функціональної вимоги до функції, яку слід порівняти пропонується використати вимогу щодо створення функції «Облік команд».

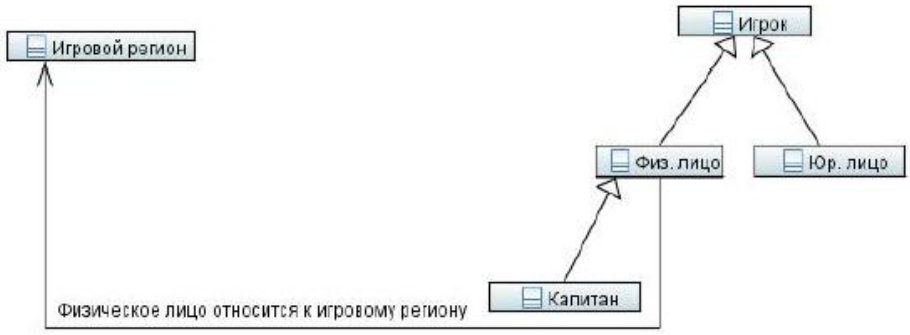

Рисунок А.1 – Подання на рівні знань вимоги до функції «Облік гравців»

Рисунок А.2 – Подання на рівні знань вимоги до функції «Облік команд»

З цих рисунків виходить, що функції можна вважати такими, що дублюють одна іншу, бо вони відрізняються 4-ма з 11 максимально можливими елементів описів цих функцій.

Під час виконання кроку 0 відбувається заповнення бази описів повторно використовуваних функцій.

Таблиця А.1 – Зміст таблиці «Requirement»

№	Значення id_reqiurement	Значення requirement_name	Значення requirement_description
1	1	Облік гравців	Облік гравців ляляля

Таблиця А.2 – Зміст таблиці «Frame»

№	Значення id_frame	Значення frame_name
1	1	Гравець
2	2	Фізична особа
3	3	Юридична особа
4	4	Капітан
5	5	Ігровий регіон

Таблиця А.3 – Зміст таблиці «Type_sign»

№	Значення id_sign	Значення sign_name
1	1	Початок зв’язку
2	2	Кінець зв’язку

Таблиця А.4 – Зміст таблиці «Communication»

№	Значення id_communication	Значення communication_name	Значення communication_type	Значення communication_power
1	1	Гравець:Фізична особа	Наслідування	1
2	2	Гравець:Юридична особа	Наслідування	1
3	3	Фізична особа: Капітан	Наслідування	1
4	4	Фізична особа: Ігровий регіон	Асоціація	1

Таблиця А.5 – Зміст таблиці «Communication_frame_type»

№	Значення id_frame	Значення id_communication	Значення id_sign
1	1	1	1
2	2	1	2
3	1	2	1
4	3	2	2
5	2	3	1
6	4	3	2
7	2	4	1
8	5	4	2

Таблиця А.6 – Зміст таблиці «Requirement_description»

№	Значення id_requirement	Значення id_frame	Значення id_communication
1	1	1	null
	1	2	null
	1	null	1
	1	3	null
	1	null	2
	1	4	null
	1	null	3
	1	5	null
	1	null	4

Під час виконання кроку 1 відбувається формування подання функціональних вимог до ІС на рівні знань.

Таблиця А.7 – Зміст таблиці «Requirement»

№	Значення id_reqiurement	Значення requirement_name	Значення requirement_description
2	2	Облік команд	Облік команд ляляля

Таблиця А.8 – Зміст таблиці «Frame»

№	Значення id_frame	Значення frame_name
1	1	Гравець
2	2	Фізична особа
3	3	Юридична особа

Кінець таблиці А.8

№	Значення id_frame	Значення frame_name
4	4	Капітан
5	5	Ігровий регіон
6	6	Команда

Таблиця А.9 – Зміст таблиці «Type_sign»

№	Значення id_sign	Значення sign_name
1	1	Початок зв’язку
2	2	Кінець зв’язку

Таблиця А.10 – Зміст таблиці «Communication»

№	Значення id_communication	Значення communication_name	Значення communication_type	Значення communication_power
1	1	Гравець:Фізична особа	Наслідування	1
2	2	Гравець:Юридична особа	Наслідування	1
3	3	Фізична особа: Капітан	Наслідування	1
4	4	Фізична особа: Ігровий регіон	Асоціація	1
5	5	Команда:Капітан	Асоціація	1
6	6	Команда: Ігровий регіон	Асоціація	1

Таблиця 4.11 – Зміст таблиці «Communication_frame_type»

№	Значення id_frame	Значення id_communication	Значення id_sign
1	1	1	1
2	2	1	2
3	1	2	1
4	3	2	2
5	2	3	1
6	4	3	2
7	2	4	1
8	5	4	2
9	6	5	1
10	4	5	2
11	6	5	1
12	5	6	2

Таблиця А.12 – Зміст таблиці «Requirement_description»

№	Значення id_requirement	Значення id_frame	Значення id_communication
1	2	1	null
2	2	2	null
3	2	null	1
4	2	3	null
5	2	null	2
6	2	4	null
7	2	null	3
8	2	6	null
9	2	null	5

Кінець таблиці А.12

№	Значення id_requirement	Значення id_frame	Значення id_communication
10	2	5	Null
11	2	null	6

Під час виконання кроку 2 відбувається формування опису IT послуги.

Опис IT послуги «Облік гравців»

Таблиця А.13 – Зміст таблиці «Requirement_description»

№	Значення id_frame	Значення id_communication
1	1	null
2	2	null
3	null	1
4	3	null
5	null	2
6	4	null
7	null	3
8	5	null
9	null	4

Опис IT послуги «Облік команд»

Таблиця А.14 – Зміст таблиці «Requirement_description»

№	Значення id_frame	Значення id_communication
1	1	null

Кінець таблиці А.14

№	Значення id_frame	Значення id_communication
2	2	Null
3	null	1
4	3	Null
5	null	2
6	4	null
7	null	3
8	6	null
9	null	5
10	5	null
11	null	6

Під час виконання кроку 3 відбувається розрахунок значення функції *Profit*(w_j).

Кількість усіх елементів S визначається кількістю записів в базі.

S = 20

Кількість унікальних елементів W визначається кількістю унікальних записів в базі.

W = 12

Значення функції вартості розраховуємо за формулою (2.9).

$$\Pr ofit(w_j) = \frac{S(w_j)}{W(w_j)} = \frac{20}{12} = 1{,}6(6)$$

Під час виконання кроку 4 відбувається розрахунок значення y_j.

Значення нелінійної активаційної функції mADALINE y_j розраховуємо за формулою (2.11): $2 - \varepsilon(k) = 2 - 0{,}1 = 1.9$; $\Pr ofit(w_j) < 2 - \varepsilon(k)$, з чого слідує, що $y_j = 0$.

Під час виконання кроку 5 умова $y_j > 0$ не виконується, тому не зберігаємо y_j.

Під час виконання кроку 6 відбувається виконання алгоритму пошуку максимально існуючої подібності.

Ми бачимо, що перше значення похибки ε(*k*) не дає знайти функціональну вимогу, яка би дублювала вимогу, що порівнюється.

Збільшуймо значення ε(*k*): ε(*k*)=ε(*k*)+0,1

Повторно виконуємо кроки 2-5

В результаті виконання цих кроків отримуємо знову y_j, що дорівнює 0.

Після збільшення похибки до 0,4 отримуємо $2-\varepsilon(k)=2-0{,}4=1{,}6$

$\Pr ofit(w_j)>2-\varepsilon(k)$, з чого виходить, що $y_j=1$.

Під час виконання кроку 7 відбувається порівняння всіх збережених у. Збережено тільки одне значення y_j, тому переходимо до кроку 8.

Під час виконання кроку 8 умова $\forall \hat{y}_j=1/2$ не виконується, отже переходимо до кроку 9.

Під час виконання кроку 9 відбувається видача масиву значень $y_j=1$ *Profit*(w_j), = 1,6(6) та $K_j^{f_{IS}}$ та завершення роботи нейрона.

З пропорції вираховуємо ступінь дублювання вимоги

Значенню *Profit*(w_j) = 2 відповідає 100%

Тоді значенню *Profit*(w_j) = 1,67 відповідає 83,5%.

Аналіз виконання роботи нейрона, показав, що він зупинився, коли знайшов вимогу у базі описів повторно використовуваних функцій, яка дублює порівнювальну вимогу на 83,5 %.