ДОДАТОК А

Програмні коди

```
import sys
from datetime import datetime, timedelta
from LinksManager import LinksManager as LM
from Storage import Storage
from DBClient import  MongoDBClient as DB

from XMLParser import  RequestStatusParser as RSP
from XMLParser import  ProjectParser
from XMLParser import  SizeParser
from XMLParser import  ActivityParser
from XMLParser import  ContributorParser
from XMLParser import  FacotidsParser
from XMLParser import AnalysisParser

import RequestRawData as Request

class ProjectCore(object):

    __savePath = "savePath\"
    __format = ".csv"

    def __init__(self):
        self.request = Request.RequestRawData()
        self.requestStatus = RSP.RequestStatusParser()

        self.parserDictionary = {
                                    "size"         :
SizeParser.SizeParser(),
                                    "project"      :
ProjectParser.ProjectParser(),
                                    "activity"     :
ActivityParser.ActivityParser(),
                                    "factoid"      :
FacotidsParser.FacotidsParser(),
                                    "analysis"     :
AnalysisParser.AnalysisParser(),
                                    "contributors" :
ContributorParser.ContributorParser()
                                  }


        self.linksManager = LM.LinksManager()
        self.db = DB.MongoDBClient()

    def GetCurrentData(self, link, parserKey):
        response = self.request.GetRawXML(link)
        isSuccess =
self.requestStatus.GetRequestStatus(response)
        if isSuccess:
            parser = self.parserDictionary[parserKey]
            data = parser.Parse(response)
            return data
```

```
        else:
            return []

    def GetHistoricalData(self, id):
        allLinks = self.linksManager.GetProjectLinks(id)
        project = self.GetCurrentData(allLinks["project"],
"project")
        if project:
            self.db.SaveNewProject(project)

    def UpdateExistingProjectData(self, projectId):
        allLinks = self.linksManager.GetProjectLinks(projectId)

        activity = self.GetCurrentData(allLinks["activity"],
"activity")
        self.db.UpdateProjectData(projectId, activity,
"activity")

        size = self.GetCurrentData(allLinks["size"], "size")
        self.db.UpdateProjectData(projectId, size, "size")

        analysis = self.GetCurrentData(allLinks["analysis"],
"analysis")
        self.db.UpdateProjectData(projectId, analysis,
"analysis")

        contributors =
self.GetCurrentData(allLinks["contributors"], "contributors")
        self.db.UpdateProjectData(projectId, contributors,
"contributors")

        factoids = self.GetCurrentData(allLinks["factoid"],
"factoid")
        self.db.UpdateProjectData(projectId, factoids,
"factoids")

    def GetProjectData(self):
        return self.db.Load("project")

    def ProcessAllProjects(self):
        allProjects = self.GetProjectData()
        for p in allProjects:
            print p["name"]
            self.SaveProjectDataToCsv(p)

    def SaveProjectDataToCsv(self, data):
        try:
            topLine = "Date,CodeLines Added,CodeLines
Removed,CommentsLines Added, CommentsLines Removed,Blanks
Added,Blanks
Removed,ManMonths,CodeLines,Comments,Blanks,TotalLines,ManMonthD
elta,Commits,Contributors \n"
```

```
            lineTemplate =
"{0},{1},{2},{3},{4},{5},{6},{7},{8},{9},{10},{11},{12},{13},{14
} \n"
            name = data["name"]
            userCount = data["userCount"]
            #title = "Name,{0},User Count,{1} \n".format(name,
userCount)
            prevManMonth = 0
            prevCommits = 0
            prevContributors = 0
            activity = data["activity"]
            sizeArray = data["size"]
            for a in activity:
                retrieveDate = a["date"]
                size =
self.FindSizeByDate(retrieveDate,sizeArray)
                totlaLines = int(size["codeLines"]) +
int(size["comments"]) + int(size["blanks"])

                manMonth = size["manMonths"]
                manMonthDelta = manMonth - prevManMonth
                prevManMonth = manMonth

                commits = a["commits"]
                contributors = a["contributors"]

                t = lineTemplate.format(retrieveDate.date(),
a["codeAdded"], a["codeRemoved"],a["commentsAdded"],
a["commentsRemoved"],a["blanksAdded"],a["blanksRemoved"],manMont
h,size["codeLines"],size["comments"],size["blanks"],totlaLines,m
anMonthDelta,commits,contributors)
                topLine += t
            f = open(self.__savePath + name + self.__format,
'w')
            f.write(topLine)
        except IOError:
            print "IO error: " + name

    def FindSizeByDate(self,activityDate,sizeArray):
          for size in sizeArray:
            sizeDate = size["date"]
            if(sizeDate == activityDate):
                return size
            else:
                continue
          return self.findNearestSize(activityDate,sizeArray)

    def findNearestSize(self,activityDate,sizeArray):
        for idx, size in enumerate(sizeArray):
            sizeDate = size["date"]
            if(sizeDate > activityDate):
                return size
        return sizeArray[len(sizeArray) - 1]
```

ДОДАТОК Б

Слайди презентації

Харківський національний університет радіоелектроніки

Атестаційна робота магістра

Дослідження моделей категорного аналізу при побудові логічних мереж

Науковий керівник
проф. Дудар З.В.

Виконав
ст.гр ІПЗм-18-4 Ковалько М.С.

Харків, 2020

1

Постановка задачі

Загальна класифікація завдань, що виникають у процесі обробки неформалізованої інформації й розв'язуваних штучним інтелектом:

- завдання аналізу (одержання з неформалізованої інформації конкретних параметрів, необхідних для застосування деякого формалізму);
- завдання нормалізації – приведення інформації до деякої еталонної форми, що особливо актуально в завданнях пошуку інформації;
- завдання синтезу – вираження внутрішнього технічного подання інформації, збереженого відповідно до формальних вимог у вигляді, адаптованому для сприйняття людиною;
- змішані завдання – найбільш загальний і трудомісткий клас завдань.

Найбільш доцільним представляється використання алгебри предикатів, системи рівнянь якої реалізуються у вигляді **логічної мережі**.

Однією з переваг такого підходу є його пряма застосовність до всіх з перерахованих у класифікації типам завдань, включаючи останній, що забезпечується декларативністю логічної мережі як методу рішення систем предикатних рівнянь.

Запис предикатних рівнянь стає можливим завдяки алгебрі скінчених предикатів.

2

Мета атестаційної роботи

- проаналізувати методи, що пов'язують теорію категорій і теорію програмної інженерії;
- проаналізувати методи представлення предикатних рівнянь за допомогою логічних мереж;
- розробити алгоритми модифікованої категорії для процесів моделювання функцій штучного інтелекту;
- розробити модель алгоритму вибору варіантів мережі;
- розробити алгоритми моделювання логічної мережі;
- розробити програмну систему, що моделює поведінку мозкоподібного комп’ютера.

- У якості проміжної галузі знань використовується алгебра скінчених предикатів, засобами якої описана предикатна інтерпретація категорії.

3

Аналіз методів створення мозкоподібних комп’ютерів

- Природно виглядає спроба досліджувати універсальний інструмент обробки неформалізованої інформації – мозок людини й змоделювати деякі його функції, що стосуються поставлених завдань.
- Підхід до моделювання функцій мозку повинен опиратися на існуючі досягнення технічних наук і математики, для того, щоб результати моделювання могли бути технічно реалізовані на базі сучасних технологій.
- Суть підходу полягає в тому, що інтелект людини розглядається як логіка в дії, як деяке матеріальне втілення механізму логіки.
- Були виконані роботи з алгебраїзації логіки.
- У результаті розроблений спеціальний математичний апарат для формульної вистави відносин і дій над ними, які називають алгебро-логічними структурами

4

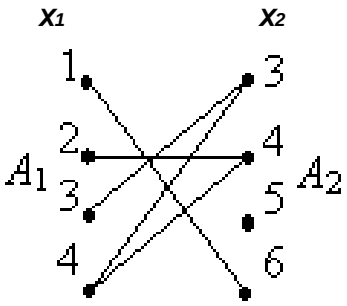
Математичний апарат теорії інтелекту

- Предикати — це основний математичний інструмент, призначений для формального опису об'єктів біоніки інтелекту.
- Важливо мати у своєму розпорядженні математичні засоби запису відносин за допомогою формул.
- Будь-яке відношення можна змістовно інтерпретувати як знання про факт, виражене деяким висловленням.
- Факт – це вичерпна характеристика дійсного стану всіх місць, що цікавлять нас.
- Знання про факт не завжди має таку визначеність, воно лише обмежує множину можливих станів.
- Нічим іншим факти й знання бути не можуть.

5

Графи відносин

- Множина і відносини взаємно однозначно пов'язані один з одним, тому їх можна не розрізняти.
- Одномісні відносини для стислості можна записувати як множину.
- Двомісні відносини зручно наочно представляти у вигляді двочасткових графів.



6

Постановка завдання дослідження

- проаналізувати переваги й недоліки існуючих методів;
- формалізувати за допомогою алгебри предикатів і побудувати логічну мережу завдання логічного розпізнавання;
- дослідити формалізацію за допомогою алгебри скінчених предикатів і моделювання логічною мережею рішення таких рівнянь;
- узагальнити завдання логічного розпізнавання й побудувати алгоритми такого моделювання;
- побудувати програмну модель логічної мережі;
- побудувати програмні моделі розглянутих завдань і досліджувати їхню роботу за допомогою програмної моделі;
- програмна модель повинна задовольняти наступним вимогам:
  - дозволяти введення й редагування моделей логічних мереж;
  - моделювати роботу логічної мережі, при необхідності – по тактах;
  - виводити на кожному такті інформацію про стан мережі;
  - допускати можливість зміни стану мережі вручну між будь-якими двома тактами.

7

Аналіз безоб'єктних категорій

- Відмінність між теорією категорій і алгеброю предикатів полягає лише в тому, що перша здійснює рух зверху вниз, націлена на пізнання вищих логічних механізмів і тому використовує в якості відправних поняття рекордного рівня спільності.
- Логічна мережа копіює дії людини, при цьому мережа діє паралельно.

  Мережа працює по тактах.
- Кожний такт поділяється на два півтакти – перший і другий.
- У першому півтакті *i*-го такту мережа для кожного зі своїх рівнянь виду *ДО*(*x*, *y)*=1 (*ДО* – відношення, що завдане таким рівнянням) відшукує:

1) по відомим знанню *Pi*(*x)* про значення змінної *x* на початку *i*-го такту знання *Q′i*(*y)* про значення змінної *b* наприкінці *i*-го такту;
2) по відомому знанню *Qi*(*y)* про значення змінної *b* на початку *i*-го такту знання *P′i*(*x)* про значення змінної *x* наприкінці *i*-го такту. Математично ці дві операції виражаються формулами:

$$x \in A(K(x, y)Pi(x))=Q'i(y);$$
$$x \in B(K(x, y)Qi(y))=P'i(x).$$

де *A* и *B* – області зміни змінних *x* и *b*.

8

У другому півтакті кожного такту

- мережа відшукує загальну частину $Pi+1(x)$ усіх знань $P'i1(x)$, $P'i2(x)$,…, $P'il(x)$ про значення кожної зі своїх предметних змінних x, що надходять по галузях мережі із усіх сторін, до полюса x.

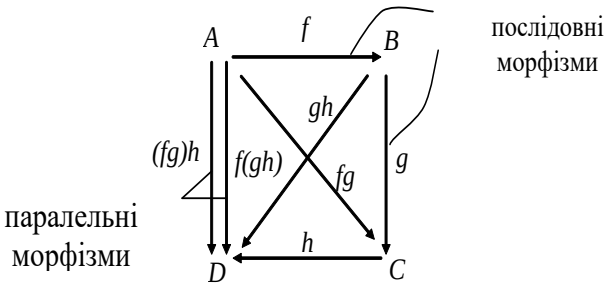
- Виражається ця операція в такий спосіб:

$$P'i1(x)\ P'i2(x)\ \ldots\ P'il(x)=Pi+1(x).$$

9

Графічне подання

- Будь-яка категорна діаграма утворюється з об'єктів і стрілок (морфізмів), являє собою орієнтований граф з розфарбованими вершинами й дугами.
- У ролі вершин графа в категорній діаграмі виступають об'єкти категорії, а в ролі дуг її морфізми



10

Властивості теорії категорій

- Властивості математичного об'єкта (простору, групи, і т.п.), які зазвичай формулюються через його внутрішню структуру, ефективно виражаються через властивості відображень цього об'єкта в однотипні з ним об'єкти.
- Саме ця можливість – переводити вивчення внутрішньої структури у вивчення зовнішніх зв'язків пояснює роль теорії категорій у вивченні системних об'єктів.

11

Алгоритм додавання проміжних змінних

В роботі запропонований алгоритм, що представляє роботу схеми по тактах:

- затримка на один такт;
- перевірка на збіг попереднього й поточного сигналів;
- передача сигналів по інтерфейсу;
- зворотний зв'язок – передача сигналів по вертикальних ланцюгах;
- формування сигналів такту;
- перевірка на збіг поточного й наступного сигналів;
- зчитування результату.

12

Бінарізація відношень

- Вихідне відношення, що містить усю інформацію і дозволяє проводити зворотні обчислення, має вигляд:
-
- Використовуючи проміжну змінну, одержано повну (розгорнуту) бінарізацію:

$$S(x,y,z) = x^0y^0z^0 \vee x^0y^1z^1 \vee x^1y^0z^1 \vee x^1y^1z^1.$$

- Результат безпосередньої бінарізації не збігається з вихідною диз'юнкцією .

$$R(x,y,z,t) = x^0y^0z^0t^0 \vee x^0y^1z^1t^1 \vee x^1y^0z^1t^2 \vee x^1y^1z^1t^3.$$

- Цим доведене, що бінарізація неповна без проміжної змінної.

Програмна модель

- Для представлення моделі завдання, що сформульовано в термінах алгебри кінцевих предикатів, у програмній моделі використовується формальна декларативна мова.
- Опис поділяється на дві частини – опис логічної мережі завдання – база даних, що містить правила, по яких можна одержати нові знання про факти, і знання про факти (база даних, що містить обмеження змінних)
- Процес опису логічної мережі також поділяється на дві частини: опис змінних і опис відносин.

Процес роботи мережі

- Перед кожним тактом користувач має можливість одержати значення всіх змінних, або додати в мережу нову інформацію, скоротивши множину, відповідні до значень змінних.
- Робота мережі триває доти, поки наприкінці кожного такту існують позначені змінні.
- Як тільки значення змінних перестають уточнюватися (мережа переходить у стабільний стан), мережа зупиняється

Можливі три різні ситуації.

1) Якщо кожна змінна прийняла конкретне значення – стан мережі відповідає рішенню заданої системи предикатних рівнянь.

2) Якщо деякі змінні зазначають множину значень – це говорить про те, що початкової інформації недостатньо для знаходження однозначного рішення і поставлене завдання допускає кілька рішень, що задовольняють поставленим умовам.

3) Випадок, коли значення всіх змінних відповідають порожнім множинам, свідчить про те, що початкові дані суперечливі.

15

Опис моделі мережі

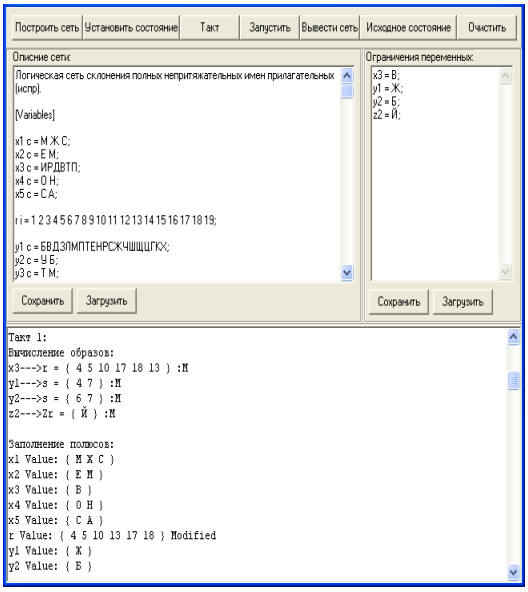
Використовується декларативна формальна мова, яка дозволяє задавати змінні (вузли) мережі й відносини - ребра.

- Головною необхідною властивістю програми є можливість одержувати й задавати стан мережі після кожного такту, що дозволяє стежити за ходом обчислень.
- У проекті наведений приклад опису логічної мережі для аналізу природної мови, що використаний в створеній програмній моделі і результати роботи моделі мережі.
- Приклади показують ефективність роботи логічної мережі в умовах поставленого завдання і застосування програмної моделі до вирішення завдань штучного інтелекту.

16

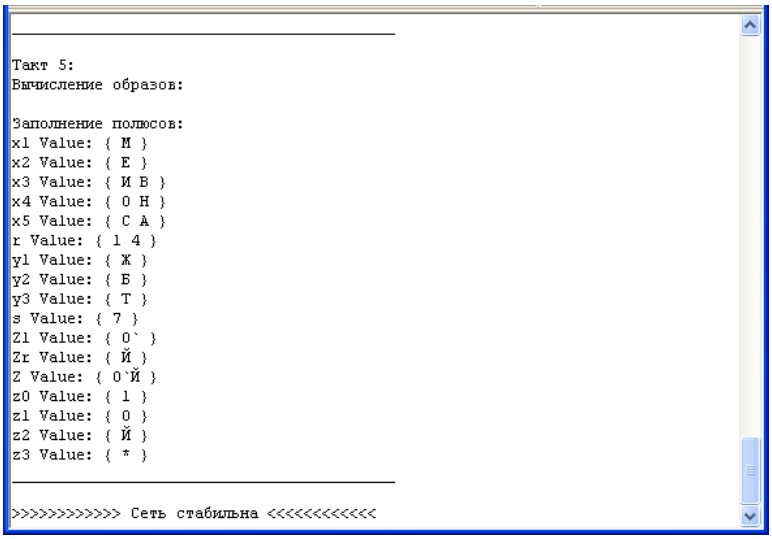
Інтерфейс програми

- надає наступні можливості:
- побудова опису моделі логічної мережі;
- завдання початкових умов для роботи мережі;
- збереження й завантаження цих описів;
- експлуатація логічної мережі – рішення систем рівнянь алгебри предикатів;
- по-тактове трасування роботи моделі для одержання наочної вистави про хід обчислень у мережі, пошуку й усунення помилок у моделі.



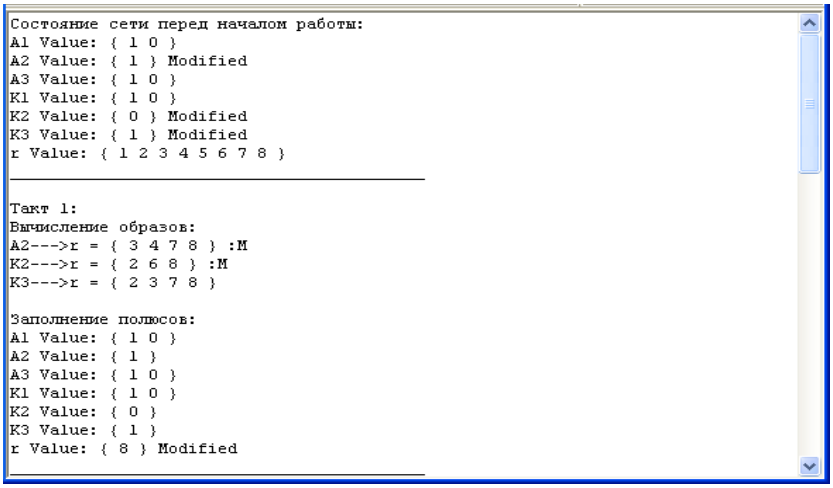
17

Завершення роботи мережі



18

Розрахунок моделі логічного розпізнавання



19

Завершення роботи моделі логічного розпізнавання

- Інтерпретація результатів в термінах вихідного завдання – при поставлених умовах завдання має єдине рішення.
- Усі три ознаки повинні бути присутніми. Крім зазначеної в умові інформації процес ДО1 обов'язково виконується. Таким чином, поставлене завдання вирішене в два такти.

20

Апробація результатів

- Підготовлено тези доповіді на
  24-й Міжнародний молодіжний форум “РАДІОЕЛЕКТРОНІКА І МОЛОДЬ В XXI СТОЛІТТІ” 2020 р.

21

Висновки

- У результаті виконання атестаційної роботи здійснене рішення ряду завдань, що відносяться до питань взаємозв'язку між теорією категорій і алгеброю предикатів, побудовано елементи програмної моделі мозкоподібної ЕОМ.
- Зроблений аналіз предметної області, докладно розглянутий обраний математичний апарат. Розглянуті можливості його застосування до деяких завдань штучного інтелекту, зокрема, до вирішення проблем обробки інформації природною мовою, завданням логічного розпізнавання й комп'ютерного зору.
- Була побудована програмна модель, що реалізує математичні моделі, побудовані засобами алгебри предикатів з результатом у вигляді логічної мережі – графа, що представляє собою систему предикатних рівнянь у графічній формі, до рішення якої й зводяться поставлені завдання.
- Проведена формалізація досліджених моделей декларативною мовою, що описана у роботі, і дозволяє будувати їх на ПК за допомогою розробленої універсальної програмної моделі логічної мережі.
- Наведено приклади рішення конкретних завдань із відповідних галузей на моделях, які застосовані в роботі, досліджений процес роботи моделей на ПК і їх ефективність.
- Зроблені висновки щодо можливостей наведеного підходу ефективно описувати й моделювати завдання з розглянутих областей.

22

ДОДАТОК В

Апробація результатів роботи

Подано тези на Міжнародний молодіжний форум

«Радіоелектроніка і молодь в XXI сторіччі

ДОСЛІДЖЕННЯ МОДЕЛЕЙ КАТЕГОРНОГО АНАЛІЗУ ПРИ ПОБУДОВІ ЛОГІЧНИХ МЕРЕЖ

Ковалько М.С.
Науковий керівник – проф. Дудар З.В.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки, 14, каф. Програмної інженерії,
тел.: (057) 702-14-46)
E-mail: maksim.kovalko@nure.ua

In the work the method of estimating the maturity (perfection) of the projects performed in the testing of software systems is developed. This method is based on TMM's five-level maturity model. Also, the use of mathematical model, the basic process of testing software data processing systems under the conditions of limited resources, as well as several methods aimed at improving the quality of software products, have been developed and substantiated.

До головних напрямків програмної інженерії відносяться задачі вдосконалення процесів життєвого циклу ПС, зокрема процесу тестування. Неухильне підвищення якості програмних продуктів – основна мета програмної інженерії та предмет турботи розробників ПС.

Відповідь на питання, як підвищити конкурентоспроможність українських програмних продуктів, як знизити ризик проектів, як досягти балансу сторін трикутника «тривалість – вартість – цілі проекту», протягом останніх років намагаються знайти відповідь не лише керівники організацій-розробників ПЗ і менеджери проектів, але і замовники, і споживачі, що використовують програмні продукти низької якості. Наслідки постачання програмних продуктів низької якості – це завжди збитки користувачів: не лише матеріальні і фінансові втрати, але і падіння престижу. Ці проблеми, в свою чергу, негативно відображаються на конкурентоспроможності організації-розробника програмних продуктів. Для забезпечення необхідного рівня якості ПС в міжнародній практиці знаходять застосування два підходи: продукто-орієнтований і процесо-орієнтований. В першому акцент робиться на оцінку якості шляхом тестування готового програмного продукту. Цей підхід базується на припущенні, що чим більше знайдено і усунено дефектів в ПС при тестуванні, тим вище його якість.

Тестування – важливий етап розроблення ПС, оскільки, з одного боку, вимагає значних витрат на проведення, а з іншого – робить великий внесок у їхню якість. Через теоретично доведену неможливість вичерпного тестування та велику потенційну вартість втрат через відмови у ПС, потрібен чітко визначений та ефективний *процес тестування*, базований на ухваленні збалансованих рішень щодо тривалості та вартості тестування для досягнення необхідного рівня довіри до якості ПС.

Методи визначення кількісних критеріїв завершення тестування та керування процесом тестування з використанням кількісних вимірів ще мало використовуються в проектах створення ПС, що призводить до того, що якість та надійність залишаються не передбачуваними. Лише за наявності достовірної та своєчасної інформації щодо стану ПС, видів ризиків і можливих втрат через відмови може бути забезпечене ефективне виконання процесу тестування.

Метою роботи є проведення комплексу досліджень з інженерії тестування ПС оброблення даних, формування ефективної стратегії тестування програмних систем, спрямованої на зниження ризику відмов під час експлуатації. Для цього в роботі розв'язуються наступні задачі:

- дослідження сучасних вимог до процесу тестування ПС;
- аналіз існуючих моделей надійності ПС, розроблення алгоритмів та програм їх реалізації;
- побудова моделі визначення оптимального часу тестування модулів ПС з урахуванням ризиків відмов;
- визначення структури базового процесу, що регламентує всі дії з підготовки, проведення та оцінювання результатів тестування, та розроблення методики виконання процесу тестування ПС оброблення даних;
- проектування та реалізація програмного комплексу підтримки інженерії тестування;
- аналіз сучасних підходів до визначення рівня зрілості процесу тестування ПС та розроблення методики оцінювання процесу тестування;
- впровадження запропонованих підходів, моделей та методик інженерії тестування в проекти з розроблення ПС оброблення даних та опис результатів випробування запропонованих моделей оцінювання оптимального часу тестування та методу оцінювання ризиків відмов програмних модулів.

З метою визначення ефективності впровадження базового процесу тестування був розроблений метод оцінювання зрілості (досконалості) виконуваних у проектах дій з тестування ПС. Цей метод грунтується на п'ятирівневій моделі зрілості TMM.

Випробування моделей мають виконуватися в конкретній інформаційно-аналітичної системі підтримки прийняття управлінських рішень, яка має складатися з програмних комплексів, об'єднаних діловим процесом оброблення даних. В ході практичної реалізації аналіз загроз відмов системи показав, що з позицій цілісності інформації найбільший внесок у ризик її відмов робить ПК контролю і введення даних до БД ORACLE, який встановлюється та одночасно функціонує на 10 робочих місцях. Для п'яти модулів цього ПК були виконані оцінки ризику відмов та часу тестування.