# ДОДАТОК А

## Лістинг фрагментів коду програми

```
          –
namespace Perst.Impl
{
 using System;
 using System.Collections;
 using System.Diagnostics;
 using Perst;

 enum Btreeresult {
 Done,
 Overflow,
 Underflow,
 Notfound,
 Duplicate,
 Overwrite
 }

 [Serializable]
 internal class Btree:Persistentcollection, Index
 {
 internal int root;
 internal int height;
 internal Classdescriptor.Fieldtype type;
 internal int nelems;
 internal bool unique;
 [Nonserialized()]
 internal int updatecounter;

 internal static int Sizeof = Objectheader.Sizeof + 4 * 4 + 1;

 internal Btree()
 {
 }

 internal Btree(byte[] obj, int offs)
 {
 height = Bytes.unpack4(obj, offs);
 offs += 4;
 nelems = Bytes.unpack4(obj, offs);
 offs += 4;
 root = Bytes.unpack4(obj, offs);
 offs += 4;
 type = (Classdescriptor.Fieldtype)Bytes.unpack4(obj, offs);
 offs += 4;
 unique = obj[offs] != 0;
 }

 internal Btree(Type cls, bool unique)
 {
 type = checktype(cls);
 this.unique = unique;
 }
```

```
 internal Btree(Classdescriptor.Fieldtype type, bool unique)
 {
 this.type = type;
 this.unique = unique;
 }

 public virtual void init(Type cls, Classdescriptor.Fieldtype[]
types, string[] fieldnames, bool unique, long autoinccount)
 {
 this.type = types[0];
 this.unique = unique;
 }

 static protected Classdescriptor.Fieldtype checktype(Type c)
 {
 Classdescriptor.Fieldtype elemtype =
Classdescriptor.converttonotnullable(Classdescriptor.gettypecode(c))
;
 if ((int)elemtype > (int)Classdescriptor.Fieldtype.tpoid
 && elemtype != Classdescriptor.Fieldtype.tparrayofbyte
 && elemtype != Classdescriptor.Fieldtype.tpdecimal
 && elemtype != Classdescriptor.Fieldtype.tpguid)
 {
 throw new
Storageerror(Storageerror.Errorcode.UNSUPPORTED_INDEX_TYPE, c);
 }
 return elemtype;
 }

 public virtual int comparebytearrays(byte[] key, byte[] item, int
offs, int length)
 {
 int n = key.Length >= length ? length : key.Length;
 for (int i = 0; i < n; i++)
 {
 int diff = key[i] - item[i+offs];
 if (diff != 0)
 {
 return diff;
 }
 }
 return key.Length - length;
 }

 public override int Count
 {
 get
 {
 return nelems;
 }
 }
```

```
public bool Isunique
{
get
{
return unique;
}
}

public int Headersize
{
get
{
return Sizeof;
}
}


public Classdescriptor.Fieldtype Fieldtype
{
get
{
return type;
}
}

public virtual Classdescriptor.Fieldtype[] Fieldtypes
{
get
{
return new Classdescriptor.Fieldtype[]{type};
}
}

public Type Keytype
{
get
{
return mapkeytype(type);
}
}

public object[] this[object from, object till]
{
get
{
return Get(from, till);
}
}

public object this[object key]
{
get
```

```
 {
 return Get(key);
 }
 set
 {
 Set(key, value);
 }
 }

 protected virtual Key checkkey(Key key)
 {
 if (key != null)
 {
 if (key.type != type)
 {
 throw                                                    new
Storageerror(Storageerror.Errorcode.INCOMPATIBLE_KEY_TYPE);
 }
 if ((type == Classdescriptor.Fieldtype.tpobject
 || type == Classdescriptor.Fieldtype.tpoid)
 && key.ival == 0 && key.oval != null)
 {
 object obj = key.oval;
 key  =  new  Key(obj,  Storage.Makepersistent(obj),  key.inclusion  !=
0);
 }
 if  (type  ==  Classdescriptor.Fieldtype.tpstring  &&  key.oval  is
string)
 {
 key  =  new  Key(((string)key.oval).Tochararray(),  key.inclusion  !=
0);
 }
 }
 return key;
 }

 public virtual object Get(Key key)
 {
 key = checkkey(key);
 if (root != 0)
 {
 Arraylist list = new Arraylist();
 Btreepage.find((Storageimpl) Storage, root, key, key, this, height,
list);
 if (list.Count > 1)
  throw new Storageerror(Storageerror.Errorcode.KEY_NOT_UNIQUE);
 }
 else if (list.Count == 0)
 {
 return null;
  else
 return list[0];
```

ДОДАТОК Б
Слайди презентації

**Міністерство освіти і науки України**
**Харківський національний університет радіоелектроніки**

**Атестаційна робота магістра**

# Дослідження алгоритмів моделювання колективної поведінки інтелектуальних агентів

**Керівник:**

проф. Шостак І.В.

**Виконав:**

ст.гр. ІПЗмзд-18-1
Вишньов С.О.

2020

1

1

## Мета роботи

**дослідження методів і алгоритмів моделювання індивідуальної й колективної поведінки інтелектуальних агентів на основі адаптивних нечітких ситуаційних мереж та комп'ютерного моделювання розроблених алгоритмів.**

2

## Узагальнена класифікація агентів

За ознакою «природні – штучні»:

За ознакою «матеріальні – віртуальні»:

По рухливості:

За ознакою «локалізовані – розподілені»:

По навченості:

- такі, яких навчають – агенти, поведінка яких враховує попередній досвід;
- ненавчальні агенти.

За ступенем можливостей представлення навколишнього середовища й засобам поведінки:

- реактивні агенти, що не мають розвиненого представлення про навколишнє середовище, механізму багатоетапних міркувань, достатніх ресурсів;
- інтелектуальні агенти, що мають такі властивості

3

## Порівняльна характеристика основних властивостей реактивних і інтелектуальних агентів

| Властивості | Реактивні агенти | Інтелектуальні агенти |
|---|---|---|
| Внутрішня модель навколишнього середовища | Примітивна | Розвинена |
| Міркування | Прості однокрокові | Складні й рефлексивні |
| Мотивація | Найпростіші реакції | Розвинена система мотивації |
| Пам'ять | Немає | Є |
| Реакція | Швидка | Повільна |
| Адаптивність | Висока | Відносно невисока |
| Модульність організації | Немає | Є |

4

## Аналіз методів моделювання поведінки інтелектуальних агентів

Найбільш актуальними є завдання моделювання поведінки інтелектуальних агентів.

Розглянуто особливості завдань моделювання поведінки інтелектуальних агентів, використавши трьох-рівневу семантичну модель «діяльність–дія–операція», що відбиває на кожному рівні як зв'язки між регулятивними компонентами поведінки агентів, так і зв’язки між базовими функціями поведінки.

У загальному випадку можна представити:

*Діяльність* = *F* (середовище, потреби, мотивація, планування, знання);

*Дія* = *G* (діяльність, об'єкти, цілі, стратегії, уміння);

*Операція* = *H* (дія, умови, завдання, тактики, навички),

5

## Основні завдання моделювання ситуативної поведінки інтелектуальних агентів

- ідентифікація поточного стану агента, станів взаємодіючих з ним агентів і навколишнього середовища;
- формування дій (керуючих рішень) і послідовностей дій залежно від виду й ситуації(й) взаємодії агентів і стратегій поведінки;
- адаптація поведінки агентів до змін умов взаємодії з іншими агентами й навколишнім середовищем з урахуванням цілей, ролей, ступеня відповідальності й ресурсів.

6

## Основні поняття нечіткого ситуаційного підходу

- нечітка ситуація - нечітка множина 2-го рівня

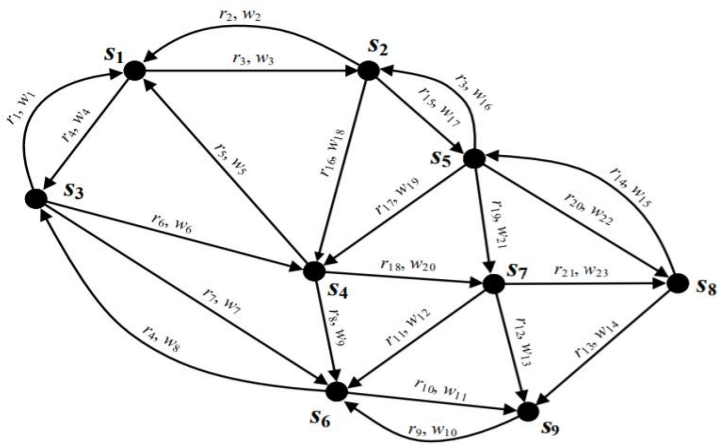
$$s = \{(\mu_s(p_i)/p_i)\}, p_i \in P$$

- нечітке керуюче рішення (дія)

$$r_k = \langle Lr_k, Tr_k, Dr_k \rangle, k \in \{1,2,\dots,K\}$$

7

## Приклад структури нечіткої ситуаційної мережі




8

## Постановка задач дослідження

У відповідності до мети роботи потрібно:

- розробити алгоритми моделювання індивідуальної й колективної поведінки ІА на основі АНСМ;
- розробити програмні засоби моделювання індивідуальної й колективної поведінки ІА на основі АНСМ;
- виконати оцінку якості прийняття рішень із використанням запропонованих алгоритмів моделювання індивідуальної і колективної поведінки ІА на основі АНСМ.

**До адаптивних нечітких ситуаційних мереж**, призначених для моделювання індивідуальної й колективної поведінки інтелектуальних агентів, пред'являються наступні основні вимоги:

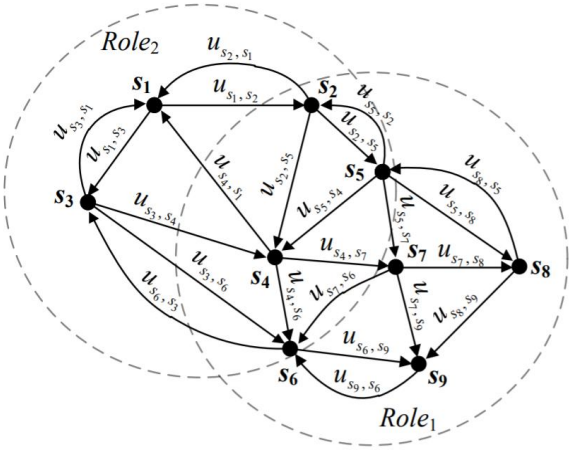
- гнучка зміна компонентів мереж (нечітких ознак, ситуацій, що управляють рішень, переходів) і їх значень для забезпечення переходів у цільові ситуації, у тому числі за умови недосяжності цільових ситуацій;
- облік залежностей між нечіткими ситуаційними ознаками при дії на них керуючих рішень;
- виконання пошуку й вибору керуючих рішень і їх послідовностей для переходів між ситуаціями АНСМ відповідно до різних критеріїв і обмеженнями при їхній побудові й адаптивній зміні.

9

## Адаптивна нечітка ситуаційна мережа

$$\text{АНСМ} = \langle P, S, R, U, Int, Roles, Adapt, Paths \rangle,$$

Приклад структури адаптивної нечіткої ситуаційної мережі




10

## Множина ситуаційних ознак

містить у собі ознаки, необхідні для аналізу взаємодії й моделювання поведінки агентів.

$$P = \{p_i\}, i = 1, \dots, I$$

- Набір ознак є загальним для всіх ролей, а ролі впливають на зміни окремих термів (значень) ознак.

11

## Ілюстрація нечіткого відношення впливу $p_i$

$\tilde{r}_{k_i}^{p_i} =$

| | $T_1^{p_i}$ | $T_2^{p_i}$ | . . . | $T_{J_i}^{p_i}$ |
|---|---|---|---|---|
| $T_1^{p_i}$ | $r_{11}$ | $r_{1,j}$ | . . . | $r_{1,J_i}$ |
| $T_2^{p_i}$ | $r_{21}$ | $r_{j,j}$ | . . . | $r_{j,J_i}$ |
| . . . | . . . | . . . | . . . | . . . |
| $T_{J_i}^{p_i}$ | $r_{J_i,1}$ | $r_{J_i,j}$ | . . . | $r_{J_i,J_i}$ |

12

## Матричне представлення відносини впливу ознаки

$$Int^{(p_i,p_z)} = \begin{array}{|c|c|c|c|c|} \hline & T_1^{p_z} & T_2^{p_z} & \dots & T_{m_z}^{p_z} \\ \hline T_1^{p_i} & Int_{11}^{(p_i,p_z)} & Int_{12}^{(p_i,p_z)} & \dots & Int_{1m_i}^{(p_i,p_z)} \\ \hline T_2^{p_i} & Int_{21}^{(p_i,p_z)} & Int_{22}^{(p_i,p_z)} & \dots & Int_{2m_i}^{(p_i,p_z)} \\ \hline \dots & \dots & \dots & \dots & \dots \\ \hline T_{m_i}^{p_i} & Int_{m_i1}^{(p_i,p_z)} & Int_{m_i1}^{(p_i,p_z)} & \dots & Int_{m_im_z}^{(p_i,p_z)} \\ \hline \end{array}$$

Для забезпечення коректного застосування керуючих рішень в умовах взаємовпливу нечітких ситуаційних ознак один на одного виконується перевірка транзитивності нечітких відносин впливу.

13

## Алгоритми побудови маршрутів

наступні випадки побудови маршрутів:

- для випадку вже сформованої мережі;
- у процесі побудови мережі;
- у випадку «розподілу» фрагментів мережі по ролях при зміні (доповненні, скороченні) структури мережі (ситуаціями, що керують рішеннями, керуючими переходами), якщо для якої-небудь ролі відсутня можливість досягнення цільової(их) ситуації(й).

$$Paths = \left\{Path^{(s_{cur},s_{targ})}\right\}$$

- Маршрути будуються для кожного агента

14

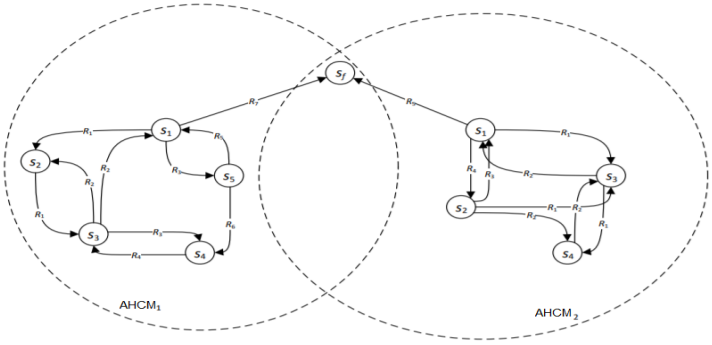
## Розроблені алгоритми

- Алгоритм зворотної побудови адаптивних нечітких ситуаційних мереж
- Алгоритм моделювання індивідуальної поведінки агентів

Запропонований алгоритм моделювання індивідуальної поведінки інтелектуальних агентів ґрунтується на АНСМ і враховує особливості поведінки як самих агентів, так і специфіку мінливого середовища, у якому вони функціонують.

15

## АНСМ, відповідна до незалежних агентів




Специфіка моделювання ситуативної поведінки агентів також впливає на вибір критерію пошуку й формування відповідної їй послідовності керуючих рішень в АНСМ.
Прикладом можуть служити обмеження для деякої ролі, для якої сумарна вага керуючих переходів до цільової ситуації не повинна перевищувати певного значення – при виборі послідовності керуючих рішень має враховуватися це обмеження.

16

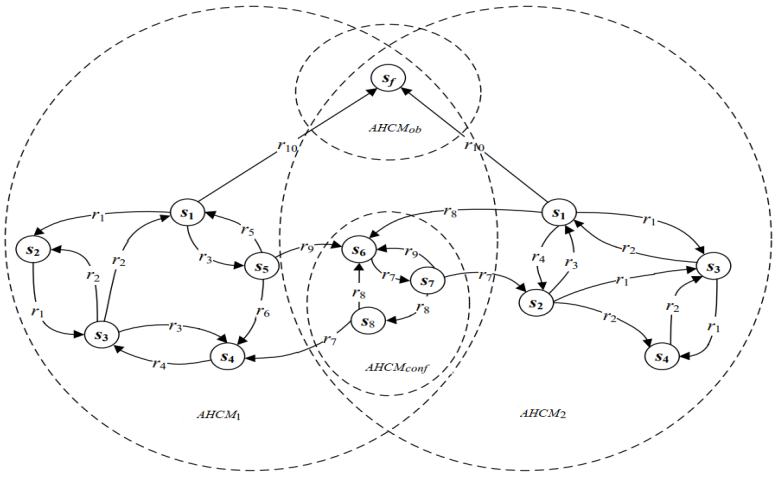
## Співробітництво агентів

характеризується наявністю у агентів загальних цілей і/або загальних ресурсів, що розподіляються.

- У випадку наявності у них тільки загальних цілей, характер взаємодії агентів аналогічний незалежності агентів із загальними цілями.
- Якщо у агентів є загальні цілі й загальні ресурси, що розподіляються, то взаємодія таких агентів аналогічна суперництву агентів.
- Необхідно враховувати, що агенти виконують дії і взаємодіють один з одним у динаміці - кожна їхня дія забирає певний час.
- Для забезпечення коректності моделювання їх поведінки з урахуванням цієї динаміки при необхідності слід здійснювати контроль часу виконання керуючих переходів.

17

## AHCM агентів, що співробітничають, із загальними цілями й ресурсами



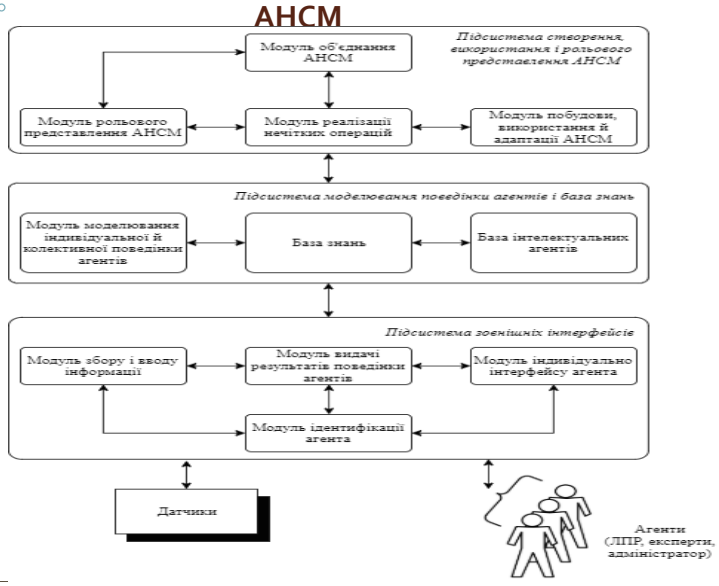

18

## Структура програмних засобів моделювання

- Вхідними даними для програми є описи компонентів АНСМ, що вводяться користувачем за допомогою спеціальних полів інтерфейсу або, що завантажуються з файлу.
- Вихідними даними є знайдені стратегії керування в мережі, представлені у вигляді текстового рядка, або збережені у файл.

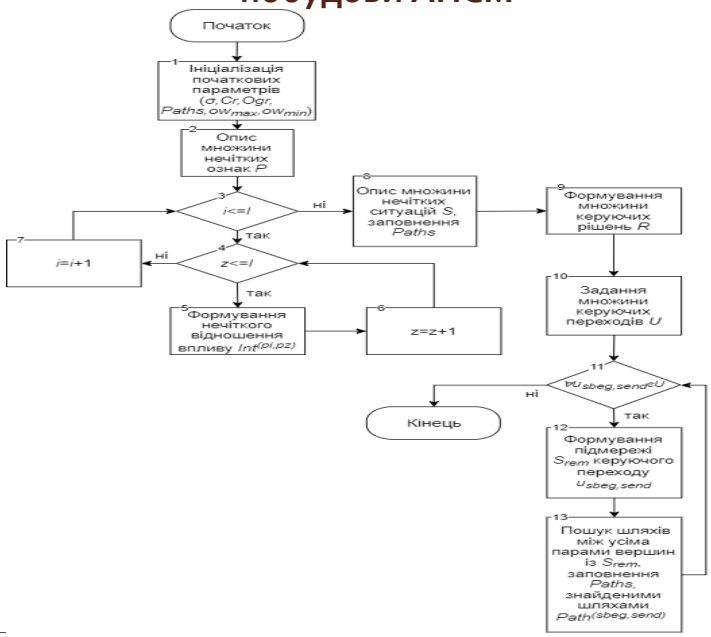
19

## Структура програмних засобів моделювання індивідуальної й колективної поведінки інтелектуальних агентів на основі АНСМ




20

## Алгоритм способу прямої побудови АНСМ

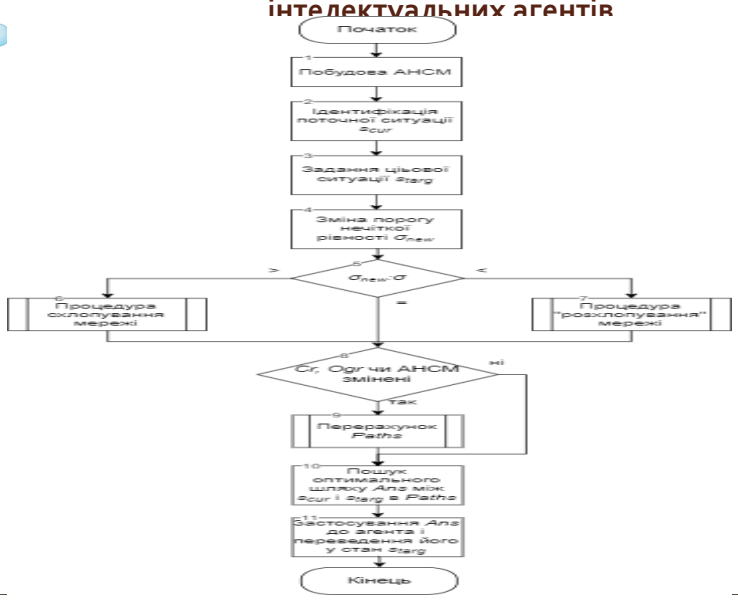



21

## Алгоритм адаптації АНСМ


22

## Схема алгоритму моделювання індивідуальної поведінки інтелектуальних агентів

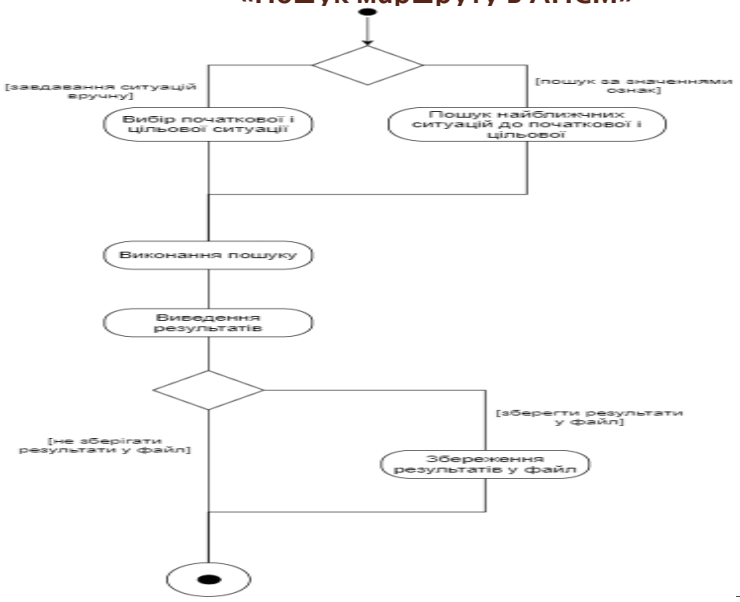



23

## Діаграма варіантів використання


24

## Діаграма діяльності для варіанту «Пошук маршруту в АНСМ»




25

## Ілюстрація роботи агентів в задачі логістичних процесів

| | Менеджер | Комірник | Начальник виробництва | Менеджер відділу закупівель |
|---|---|---|---|---|
| Менеджер | Конкуренція | Співробітництво | Співробітництво | Незалежність |
| Комірник | Співробітництво | Співробітництво | Співробітництво | Співробітництво |
| Начальник виробництва | Співробітництво | Співробітництво | – | Незалежність |
| Менеджер відділу закупівель | Незалежність | Співробітництво | Незалежність | Співробітництво |

26

# Висновки

**В результаті виконання атестаційної роботи розроблено:**

Запропоновано алгоритм моделювання індивідуальної поведінки інтелектуальних агентів, що відрізняється:

• застосуванням адаптивних нечітких ситуаційних мереж;

• пошуком і вибором послідовностей керуючих рішень для переходу агента з поточної ситуації в цільову, що дозволяє підвищити якість підтримки прийняття рішень, заснованих на результатах такого моделювання.

Розроблено алгоритми, що реалізують запропоновані способи побудови АНСМ і моделі індивідуальної й колективної поведінки інтелектуальних агентів, включаючи алгоритми:

• прямої й зворотної побудови АНСМ;

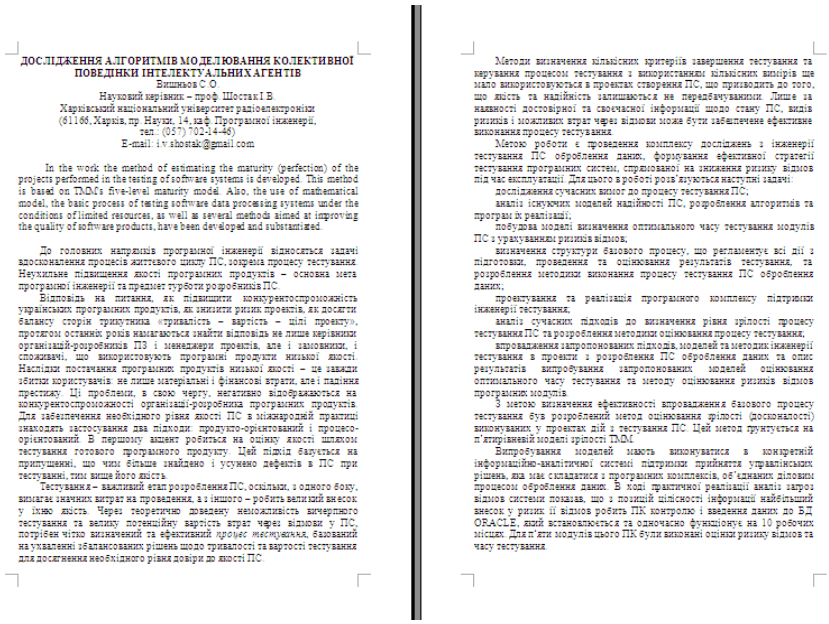
• адаптації АНСМ, пошуку й ідентифікації поточної ситуації в АНСМ, об'єднання АНСМ агентів;

• моделювання індивідуальної й колективної поведінки агентів.

Розроблено прототип програмного засобу моделювання індивідуальної й колективної поведінки інтелектуальних агентів на основі АНСМ, наведено практичний приклад використання в якості підтримки логістичних рішень.

# ДОДАТОК В

Апробація результатів роботи



**ДОСЛІДЖЕННЯ АЛГОРИТМІВ МОДЕЛЮВАННЯ КОЛЕКТИВНОЇ ПОВЕДІНКИ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ**

Вишньов С.О.
Науковий керівник – проф. Шостак І.В.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки, 14, каф. Програмної інженерії,
тел.: (057) 702-14-46)
E-mail: i.v.shostak@gmail.com

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 відноситься 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 ухвалених 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 ґрунтується 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 запрос 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.

Подано тези на Міжнародний молодіжний форум «Радіоелектроніка і молодь в XXI сторіччі»