ДОДАТОК А

Лістинг фрагментів коду програми

```
#ifndef __CONFIG_H
#define __CONFIG_H

#ifdef __APPLE__
#include <Availabilitymacros.h>
#endif

/* Define redis_fstat to fstat or fstat64() */
#if defined(__APPLE__) && !defined(MAC_OS_X_VERSION_10_6)
#define redis_fstat fstat64
#define redis_stat stat64
#else
#define redis_fstat fstat
#define redis_stat stat
#endif

/* Test for proc filesystem */
#ifdef __linux__
#define HAVE_PROC_STAT 1
#define HAVE_PROC_MAPS 1
#define HAVE_PROC_SMAPS 1
#endif

/* Test for task_info() */
#if defined(__APPLE__)
#define HAVE_TASKINFO 1
#endif

/* Test for backtrace() */
#if defined(__APPLE__) || defined(__linux__)
#define HAVE_BACKTRACE 1
#endif

/* Test for polling API */
#ifdef __linux__
#define HAVE_EPOLL 1
#endif

#if   (defined(__APPLE__)   &&   defined(MAC_OS_X_VERSION_10_6))   ||
defined(__Freebsd__) || defined(__Openbsd__) || defined (__Netbsd__)
#define HAVE_KQUEUE 1
#endif

#ifdef __sun
#include <sys/feature_tests.h>
#ifdef _DTRACE_VERSION
#define HAVE_EVPORT 1
#endif
#endif

/* Define aof_fsync to fdatasync() in Linux and fsync() for all the
rest */
```

```
#ifdef __linux__
#define aof_fsync fdatasync
#else
#define aof_fsync fsync
#endif

/* Define rdb_fsync_range to sync_file_range() on Linux, otherwise
we use
 * the plain fsync() call. */
#ifdef __linux__
#include <linux/version.h>
#include <features.h>
#if defined(__GLIBC__) && defined(__GLIBC_PREREQ)
#if (LINUX_VERSION_CODE >= 0x020611 && __GLIBC_PREREQ(2, 6))
#define HAVE_SYNC_FILE_RANGE 1
#endif
#else
#if (LINUX_VERSION_CODE >= 0x020611)
#define HAVE_SYNC_FILE_RANGE 1
#endif
#endif
#endif

#ifdef HAVE_SYNC_FILE_RANGE
#define                                    rdb_fsync_range(fd,off,size)
sync_file_range(fd,off,size,SYNC_FILE_RANGE_WAIT_BEFORE|SYNC_FILE_RA
NGE_WRITE)
#else
#define rdb_fsync_range(fd,off,size) fsync(fd)
#endif

/* Check if we can use setproctitle().
 * BSD systems have support for it, we provide an implementation for
 * Linux and osx. */
#if   (defined   __Netbsd__   ||   defined   __Freebsd__   ||   defined
__Openbsd__)
#define USE_SETPROCTITLE
#endif

#if (defined __linux || defined __APPLE__)
#define USE_SETPROCTITLE
#define INIT_SETPROCTITLE_REPLACEMENT
void spt_init(int argc, char *argv[]);
void setproctitle(const char *fmt, ...);
#endif

/* Byte ordering detection */
#include <sys/types.h> /* This will likely define BYTE_ORDER */

#ifndef BYTE_ORDER
#if (BSD >= 199103)
# include <machine/endian.h>
#else
```

```
#if defined(linux) || defined(__linux__)
# include <endian.h>
#else
#define   LITTLE_ENDIAN  1234 /*  least-significant  byte  first  (vax,
pc) */
#define   BIG_ENDIAN     4321 /*  most-significant  byte  first  (IBM,
net) */
#define   PDP_ENDIAN     3412 /* LSB first in word, MSW first in long
(pdp)*/

#if  defined(__i386__)  ||  defined(__x86_64__)  ||  defined(__amd64__)
|| \
   defined(vax) || defined(ns32000) || defined(sun386) || \
   defined(MIPSEL) || defined(_MIPSEL) || defined(BIT_ZERO_ON_RIGHT)
|| \
   defined(__alpha__) || defined(__alpha)
#define BYTE_ORDER    LITTLE_ENDIAN
#endif

#if   defined(sel)    ||    defined(pyr)    ||    defined(mc68000)    ||
defined(sparc) || \
    defined(is68k)    ||    defined(tahoe)    ||    defined(ibm032)    ||
defined(ibm370) || \
    defined(MIPSEB)   ||   defined(_MIPSEB)   ||   defined(_IBMR2)   ||
defined(DGUX) ||\
    defined(apollo) || defined(__convex__) || defined(_CRAY) || \
    defined(__hppa) || defined(__hp9000) || \
    defined(__hp9000s300) || defined(__hp9000s700) || \
    defined (BIT_ZERO_ON_LEFT) || defined(m68k) || defined(__sparc)
#define BYTE_ORDER  BIG_ENDIAN
#endif
#endif /* linux */
#endif /* BSD */
#endif /* BYTE_ORDER */

/* Sometimes after including an Os-specific header that defines the
 * endianess we end with __BYTE_ORDER but not with BYTE_ORDER that
is what
 * the Redis code uses. In this case let's define everything without
the
 * underscores. */
#ifndef BYTE_ORDER
#ifdef __BYTE_ORDER
#if defined(__LITTLE_ENDIAN) && defined(__BIG_ENDIAN)
#ifndef LITTLE_ENDIAN
#define LITTLE_ENDIAN __LITTLE_ENDIAN
#endif
#ifndef BIG_ENDIAN
#define BIG_ENDIAN __BIG_ENDIAN
#endif
#if (__BYTE_ORDER == __LITTLE_ENDIAN)
#define BYTE_ORDER LITTLE_ENDIAN
#else
```

```
#define BYTE_ORDER BIG_ENDIAN
#endif
#endif
#endif
#endif

#if !defined(BYTE_ORDER) || \
    (BYTE_ORDER != BIG_ENDIAN && BYTE_ORDER != LITTLE_ENDIAN)
     /* you must determine what the correct bit order is for
      * your compiler - the next line is an intentional error
      * which will force your compiles to bomb until you fix
      * the above macros.
      */
#error "Undefined or invalid BYTE_ORDER"
#endif

#if (__i386 || __amd64) && __GNUC__
#define GNUC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 +
__GNUC_PATCHLEVEL__)
#if GNUC_VERSION >= 40100
#define HAVE_ATOMIC
#endif
#endif

#endif
```

ДОДАТОК Б

Слайди презентації

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Атестаційна робота магістра

Дослідження алгоритмів непараметрічного керування в багатомірних системах

Керівник: проф. Шостак І.В.
Виконав: ст. гр. ІПЗмзд-18-1 Панфьоров О.Т.

1

Об'єкт дослідження

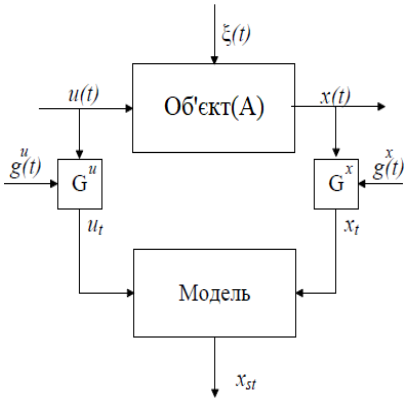
Основна мета роботи полягає в побудові й дослідженні непараметричних моделей і алгоритмів керування для багатомірних дискретно-безперервних процесів

В даній роботі досліджується новий клас процесів, названих «трубчастими» (або Н-процесами).

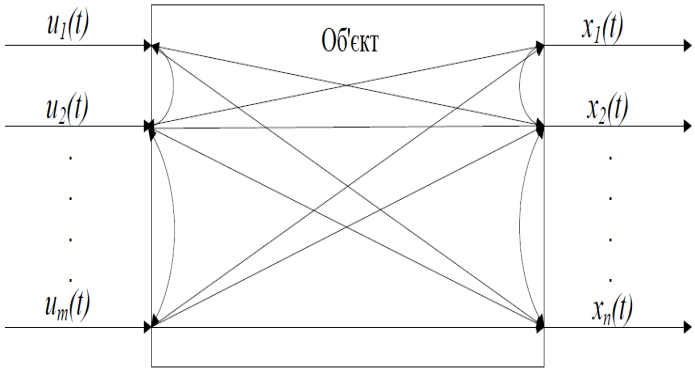
Ці процеси були помічені при моделюванні технологічних процесів у металургії. Було виявлено, що компоненти вектору входу досліджуваного процесу зв'язані стохастичною залежністю, внаслідок чого він протікає не у всій області, встановленої технологічним регламентом підприємства, а лише в деякій його підобласті

2

Загальна схема досліджуваного процесу



Зв'язки між змінними процесу



3

Рівні апріорної інформації

- **байєсів рівень апріорної інформації**. З точністю до параметрів відомі: параметрична модель досліджуваного об'єкта, закони розподілу випадкових перешкод і рівняння каналів зв'язків. Необхідно оцінити параметри параметричної моделі об'єкта;

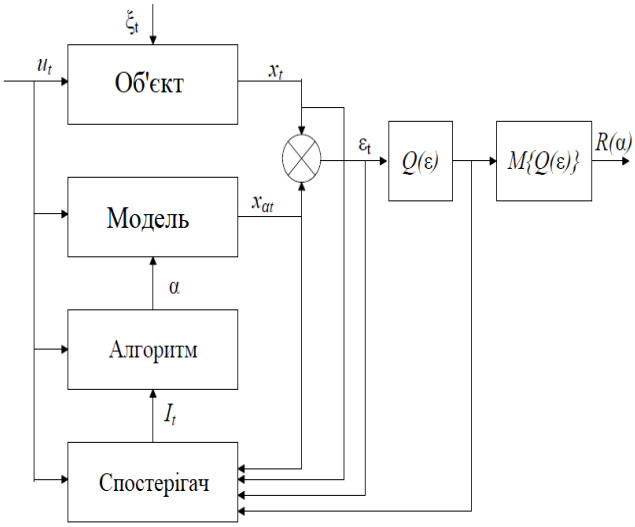
- **рівень параметричної невизначеності**. Параметрична модель об'єкта дослідження відома з точністю до параметрів, які необхідно оцінити. Відомі деякі характеристики випадкових перешкод. Вирішується завдання ідентифікації в «вузькому» змісті;

- **рівень непараметричної невизначеності**. На цьому рівні апріорної інформації відсутній етап визначення параметричної структури досліджуваного об'єкта, тому вимоги до рівня апріорної інформації слабшають, але тут потрібна інформація якісного характеру (однозначність або неоднозначність характеристик, лінійність процесу або характер його нелінійності й ін.). Для рішення завдання ідентифікації в цьому випадку застосовують методи непараметричної статистики;

- **рівень параметричної й непараметричної невизначеності**. Це випадок, коли завдання ідентифікації багато зв'язної системи формулюється в умовах і параметричної, і непараметричної апріорної інформації. Моделі тут являють собою взаємозалежну систему параметричних і непараметричних співвідношень.

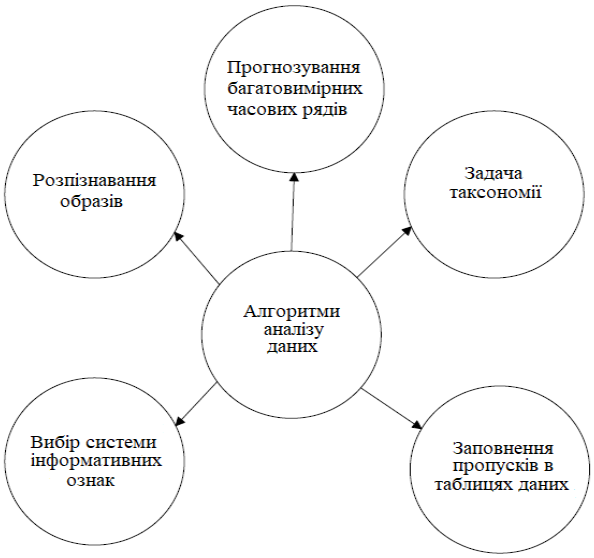
4

Схема завдання ідентифікації



5

Класифікація завдань аналізу даних



6

Завдання відновлення пропусків «вхідних-вихідних» змінних у матрицях спостережень

- Вирішується в умовах малої апріорної інформації.
- Актуальною є методика відновлення пропусків, заснована на непараметричних алгоритмах оцінки функції регресії.
- Застосування запропонованої методики дозволяє підвищити точність рішення завдання ідентифікації.
- На точність відновлення впливають: обсяг вихідної вибірки, перешкоди, що діють у каналах вимірів, кількість пропусків у даних і якісні властивості досліджуваного процесу. Природно, що при малих обсягах вибірки, при наявності великої кількості пропусків ефективність застосування методики знижується.
- Викиди в даних також впливають на точність рішення завдання ідентифікації. Існуючі робастні алгоритми дозволяють лише послабити вплив викиду на результати оцінювання.
- Більшість існуючих методів видалення викидів вимагають нормального розподілу даних.
- У цьому зв'язку потрібно **запропонувати алгоритми виключення викидів з вихідної вибірки** спостережень, що заснована на непараметричній оцінці функції регресії за спостереженнями.
- Дана методика дозволяє виключати з вибірки спостережень викиди. Обчислювальні експерименти показали, що пропонована методика дозволяє підвищити точність рішення завдання ідентифікації.

7

Постановка задач дослідження

досліджувати непараметричну методику відновлення пропусків «вхідних-вихідних» змінних матриці спостережень;

- проаналізувати непараметричну методику виключення викидів з вихідної матриці спостережень змінних процесу;
- розробити й досліджувати модифікований параметричний алгоритм ідентифікації для побудови моделей дискретно-безперервних безінерційних процесів «трубчастого» типу;
- дослідити непараметричний алгоритм дуального керування багатомірним безінерційним об'єктом із запізнюванням;

При виконанні роботи мають використовуватися методи параметричної теорії ідентифікації, непараметричної теорії ідентифікації, теорії керування, теорії адаптивних та систем, що навчаються, математичної статистики:

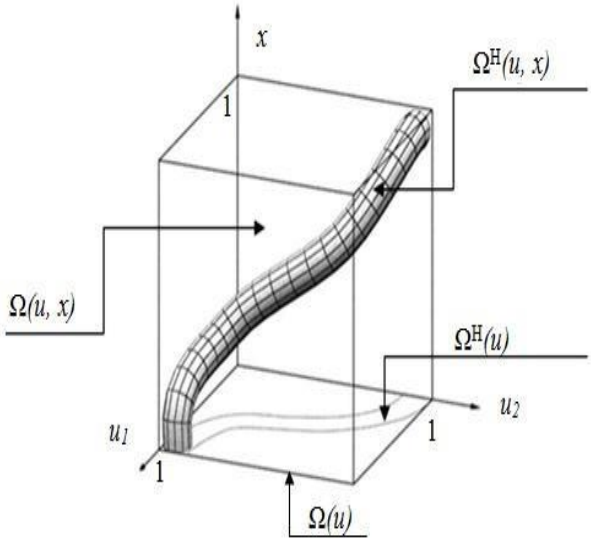
- проаналізувати математичну модель, що заснована на модифікації параметричного алгоритму ідентифікації процесів, що мають «трубчасту» структуру в просторі «вхідних-вихідних» змінних
- запропонувати модифікацію алгоритму дуального керування дискретно-безперервними процесами «трубчастого» типу. Особливість даних моделей полягає в тому, що при керуванні багатомірним об'єктом кожний компонент вектора керуючого впливу формується з урахуванням значень попередніх компонентів, що підвищує точність керування.

8

Об'єкт із «трубчастою» структурою

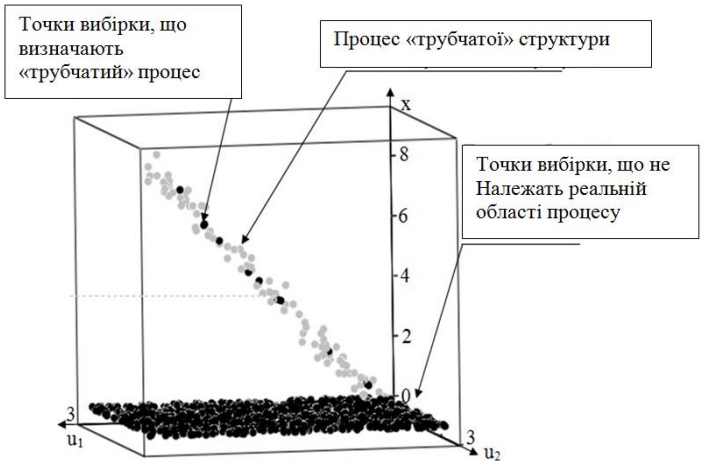
точки процесу розташовуються не по усієї площі
одиничного квадрата Ω(*u*), а лише в його області **Ω *н*(*u*)**

області Ω*H*(*u*) дослідникові завжди невідомі, оскільки її розміри й положення визначаються видом зв'язків між змінними процесу, про які немає певної інформації.



9

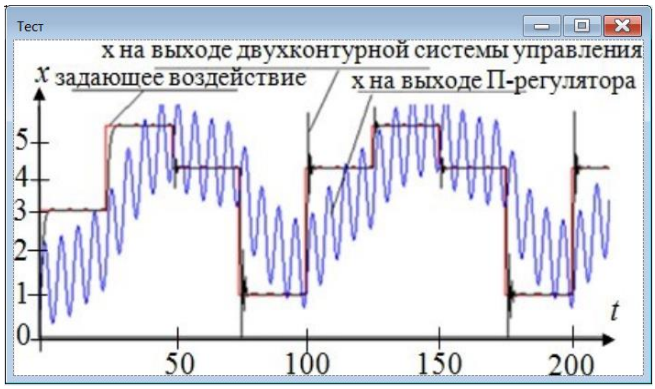
Модифікована параметрична модель «трубчастого» процесу з використанням оцінки індикаторної функції



10

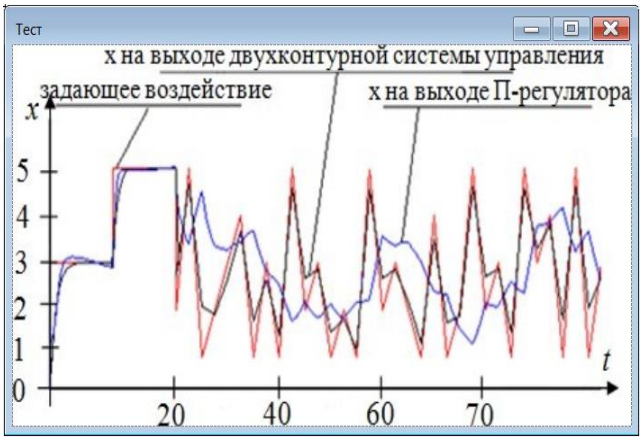
Розрахунок варіантів функціонування

об'єкт описується рівнянням: $x(t) = 2u^2(t) + 1.5u(t) + \mu(t) + g^x(t).$



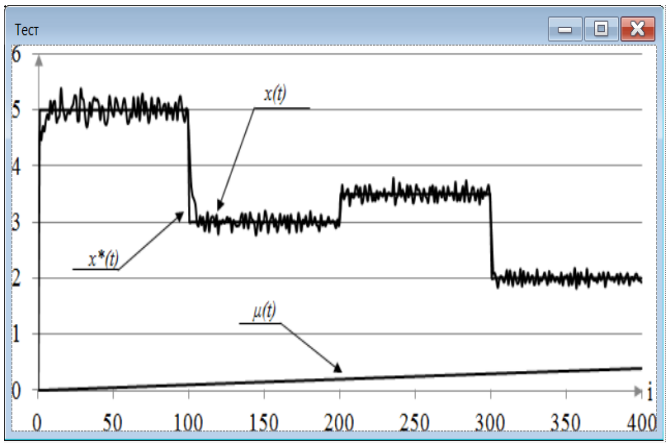
11

Результат роботи системи при випадковому завданні



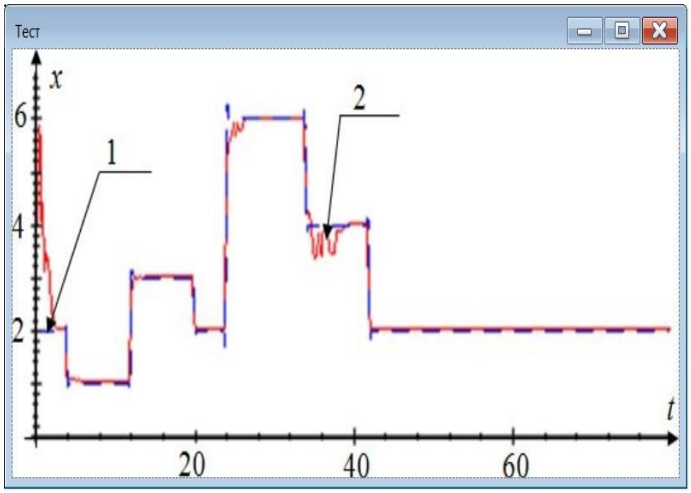
12

Результати обчислювального експерименту при перешкоді в 5%



13

Непараметричне керування процесом «трубчастого» типу



З екранної форми видно, що непараметричний регулятор досить добре справляється з поставленим завданням

14

Висновки

- Розглянуте завдання аналізу даних, що містять у собі пропуски й викиди.
- Результати обчислювальних експериментів показали, що завдання ідентифікації по заповненій за допомогою пропонованого алгоритму матриці вирішується більш точно, ніж по вихідній матриці спостережень із пропусками.
- При виключенні викиду з вибірки спостережень точність рішення завдання ідентифікації, як показують обчислювальні експерименти, підвищується.
- У роботі розглядається проблема дуального керування багатомірними системами в умовах непараметричної невизначеності.

- Пропонується модифікований алгоритм непараметричного дуального керування, що продемонстрував свою ефективність у ряді обчислювальних експериментів.
- Особливістю непараметричних алгоритмів дуального керування є те, що при керуванні багатомірним об'єктом кожний компонент вектора керуючого впливу формується з урахуванням значень попередніх компонентів, що значно підвищує точність керування.
- Алгоритм використовує схему керування процесом, що включає в себе зовнішній контур керування, який дозволить більш якісно вести технологічний процес.

ДОДАТОК В

Апробація результатів роботи

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

ДОСЛІДЖЕННЯ АЛГОРИТМІВ НЕПАРАМЕТРІЧНОГО КЕРУВАННЯ В БАГАТОМІРНИХ СИСТЕМАХ

Панфьоров О.Т.
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 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 завдяки збитки користувачів не лише матеріальні і фінансові втрати, але і падіння престижу. Ці проблеми, в свою чергу, негативно відображаються на конкурентоспроможності організацій-розробників 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 розробки 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 ухвалені 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 обробляння 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 обробляння 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 обробляння 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 обробляння даних. В ході практичної реалізації аналізу натури відмов системи показав, що з появи 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.