ДОДАТОК А

Фрагмент лістингу вихідних кодів ПЗ

Фрагмент лістингу вихідних кодів ПЗ

```
Worker.cs
using System;
using System.Collections.Generic;
using System.Threading;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace Network
{
  public enum Workermodes
  {
    Client, Server, Commonserverservice, Commonclientservice
  }

  [Serializable]
  public abstract class Worker
  {
    public bool workfinished = false;
    protected Thread _workthread;
    protected Ipendpoint _remoteep;
    protected Ipendpoint _localep;
    public abstract void Free();
    public abstract void Connect();
    public abstract void Connect(Ipendpoint Remoteep);
    public abstract void Bind();
  }
}


Udpworkerfields.cs
using System;
using System.Collections.Generic;
using System.Net.Sockets;
using System.Text;

namespace Network
{
  public partial class Udpworker:Worker
  {
    Socket _udpsocket;
    int _packetsize = 100;
    int _maxsendingattempts = 5;
  }
}


Udpworker.cs
using System;
```

```
using System.Collections.Generic;
using System.Threading;
using System.Text;
using System.Net.Sockets;
using System.Net;

namespace Network
{
  public partial class Udpworker:Worker
  {
      public Udpworker(Ipendpoint Localep, Ipendpoint Remoteep, int
Receivetimeout, int Packetsize)
    {
              _udpsocket = new Socket(Addressfamily.Internetwork,
Sockettype.Dgram, Protocoltype.Udp);
      _udpsocket.Receivetimeout = Receivetimeout;
      _packetsize = Packetsize;
      _remoteep = Remoteep;
      _localep=Localep;
                  _udpsocket=new  Socket(Addressfamily.Internetwork,
Sockettype.Dgram, Protocoltype.Udp);
    }

    public override void Bind()
    {
      _udpsocket.Bind(_localep);
    }

    public override void Connect()
    {
      _udpsocket.Connect(_remoteep);
    }

    public override void Connect(Ipendpoint Remoteep)
    {
      _remoteep = Remoteep;
      _udpsocket.Connect(_remoteep);
    }

    public void Senddata(byte[] databuffer, bool Ignoretimeout)
    {
      int currentposinpacket = 0;
      int packetindex = 0;
      int temp = (int)(databuffer.Length / _packetsize);
        int packetcount = (databuffer.Length % _packetsize == 0) ?
temp : temp + 1;
                                    byte[]      packetheader      =
Encoding.Default.Getbytes(packetindex.Tostring()
+"/"+packetcount.Tostring() + " ");
      byte[] packet = new byte[_packetsize];
                  Buffer.Blockcopy(packetheader,   0,   packet,   0,
packetheader.Length);
      currentposinpacket += packetheader.Length;
```

```
      for (int i = 0; i < databuffer.Length; i++)
      {
        packet[currentposinpacket] = databuffer[i];
        currentposinpacket++;
               if ((currentposinpacket >= _packetsize) || (i ==
databuffer.Length - 1))
        {
          int attemptscount = 0;
          while (!sendpacket(packet,packetindex,Ignoretimeout))
          {
            attemptscount++;
            if (attemptscount >= _maxsendingattempts)
            {
                    throw new Applicationexception("Can't send data.
Client not response.");
            }
          }
                                                    packetheader     =
Encoding.Default.Getbytes(packetindex.Tostring()     +     "/"     +
packetcount.Tostring() + " ");
          currentposinpacket = 0;
          packetindex++;
          packet = new byte[_packetsize];
                     Buffer.Blockcopy(packetheader,  0,  packet,  0,
packetheader.Length);
          currentposinpacket += packetheader.Length;
        }
      }
    }

    public byte[] Receivedata(bool Ignoretimeout)
    {
      int bytestoencode = (_packetsize > 20) ? 20 : _packetsize;
      byte[] packet = new byte[_packetsize];
      _udpsocket.Receive(packet);
      if (!sendpacketconfirm(packet)) return null;
       string receivedmessage = Encoding.Default.Getstring(packet, 0,
bytestoencode);
          int packetscount = 0; int currentpacketindex = 0; int
packetsreceived = 0;
                                                                  if
(int.Tryparse(receivedmessage.Substring(receivedmessage.Indexof('/')
+ 1,
                                  receivedmessage.Indexof('    ')    -
(receivedmessage.Indexof('/') + 1)), out packetscount))
      {
        List<byte[]> packets = new List<byte[]>(packetscount);
        packets.Add(packet);
        packetsreceived++;
        while (packetsreceived < packetscount)
        {
          _udpsocket.Receive(packet);
          if (!sendpacketconfirm(packet)) return null;
```

```
          packets.Add(packet);
          packetsreceived++;
        }
        List<byte> result = new List<byte>();
        currentpacketindex = 0;
        while (currentpacketindex < packetscount)
        {
          packet = packets.Find(delegate(byte[] p)
          {
                  receivedmessage = Encoding.Default.Getstring(p, 0,
bytestoencode);
                  int index = int.Parse(receivedmessage.Substring(0,
receivedmessage.Indexof('/')));
            return index == currentpacketindex;
          });
          if (packet != null)
          {
            packet = removeudppacketheader(packet);
            result.Addrange(packet);
            currentpacketindex++;
          }
          else
          {
              packet = receivepacket(currentpacketindex.Tostring() +
"/" + packetscount.Tostring(), bytestoencode,Ignoretimeout);
            sendpacketconfirm(packet);
            packet = removeudppacketheader(packet);
            result.Addrange(packet);
            currentpacketindex++;
          }
        }
        return result.Toarray();
      }
```

ДОДАТОК Б

Слайди презентації

Міністерство освіти і науки України

Атестаційна робота магістра

Дослідження алгоритмічних засобів забезпечення мережевої безпеки та веб-сайтів

Керівник, проф. Шостак І.В
Виконав, ст.гр. ІПЗм-18-4 Кочегура Д. В.

1

Мета роботи

- Метою дослідження є розробка рекомендацій з поліпшення продуктивності програмних засобів, зокрема мережевих екранів, що використовуються для захисту даних інформаційних веб-сайтів.

2

Види вразливостей веб-сайтів

- відсутність перевірки параметрів в HTTP-запитах.
- недотримання політик керування доступом до ресурсів.
- недотримання правил керування обліковими записами й користувацькими сесіями.
- вразливості, пов'язані з помилками в механізмі Cross-Site Scripting, використовуваному для перенаправлення користувача на інші сайти.
- помилки переповнення буфера, наявні в багатьох програмних продуктах.
- відсутність належного контролю над параметрами, переданими комп'ютерами при доступі до зовнішніх ресурсів.
- невдале використання криптографії.
- відсутність належного захисту підсистем вилученого адміністрування.

3

Рішення про надання доступу

- ідентифікатор суб'єкта (ідентифікатор користувача, мережеву адресу комп'ютера, тощо). – подібні ідентифікатори є основою добровільного управління доступом;
- атрибути суб'єкта (мітка безпеки, група тощо), мітки безпеки – основа примусового управління доступом;
- місце дії (системна консоль, надійний вузол мережі, тощо);
- час дії (більшість дій доцільно вирішувати тільки в робочий час);
- внутрішні обмеження сервісу (число користувачів згідно ліцензії на програмний продукт, тощо).

4

Екранування

- дозволяє підтримувати доступність сервісів внутрішньої області, зменшуючи або взагалі ліквідуючи навантаження, індуковану зовнішньою активністю.
- зменшується вразливість внутрішніх сервісів безпеки, оскільки спочатку сторонній зловмисник повинен подолати екран, де захисні механізми сконфігуровані особливо ретельно і жорстко.
- система, на відміну від універсальної, може бути влаштована більш простим і, отже, більш безпечним чином.
- екранування дає можливість контролювати інформаційні потоки, спрямовані в зовнішню область, що сприяє підтримці режиму конфіденційності.

5

Постановка задач дослідження

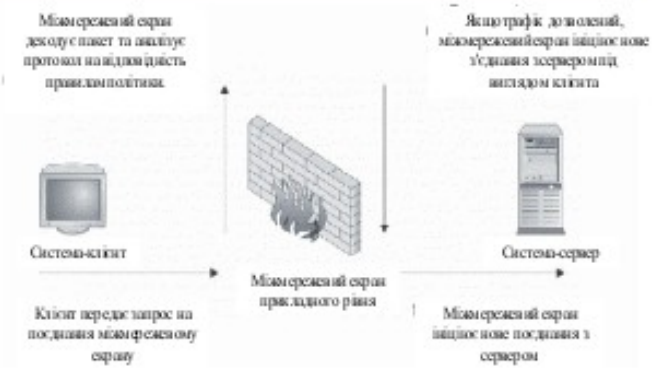
Для досягнення поставленої мети планується розв'язати наступні завдання:

- провести аналіз стану проблеми забезпечення мережної безпеки програмними засобами, зокрема мережевими екранами, на інформаційних веб-сайтах;
- провести планування експерименту по визначенню продуктивності системи мережевих екранів, включених у мережу за типовою схемою захисту даних інформаційних веб-сайтів;
- розробити прототип програмного забезпечення для проведення експерименту;
- одержати емпіричні формули залежності продуктивності досліджуваних мережевих екранів від використовуваних правил безпеки;
- проаналізувати отримані емпіричні формули й розробити рекомендації з підвищення продуктивності мережевих екранів у заданих умовах.

- У ході роботи планується перевірити гіпотезу про те, що час, що витрачений на перевірку пакетів на відповідність прийнятій політиці безпеки, можна оптимізувати шляхом перерозподілу порядку правил безпеки в списках мережевого екрана таким чином, щоб перевірки, що найбільше повільно виконуються, проводилися як можна рідше.

6

Класифікація міжмережевих екранів



З'єднання модуля доступу міжмережевого екрана прикладного рівня

7

Передача трафіку через міжмережевий екран з фільтрацією пакетів

8

Фактори експерименту по виміру продуктивності зовнішнього МЕ

Таблиця 2.5 –

№ фактора	1	2	3	4	5
№ правил	1,2,3	4,5	13,16,18	12,14,15	17

Перші 3 правила обробляють трафік між демілітаризованою зоною (ДМЗ) і

9

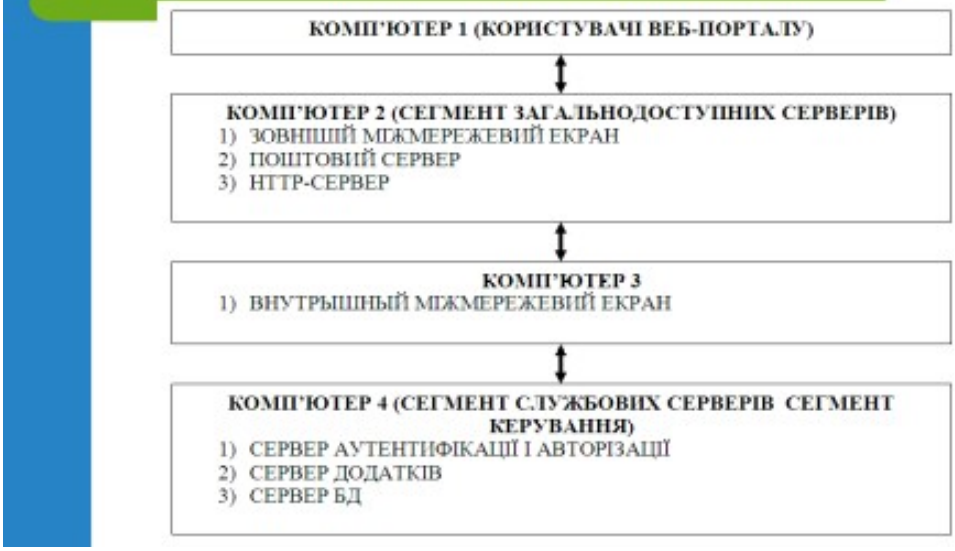
Фактори експерименту по виміру продуктивності внутрішнього мережевого екрану

№ фактора	1	2	3
№ правил	6, 7	8, 9	10, 11

Усі пари правил, що представлено у таблиці дозволяють вхідний і вихідний трафік одного типу, тому логічно представити їх як один фактор.

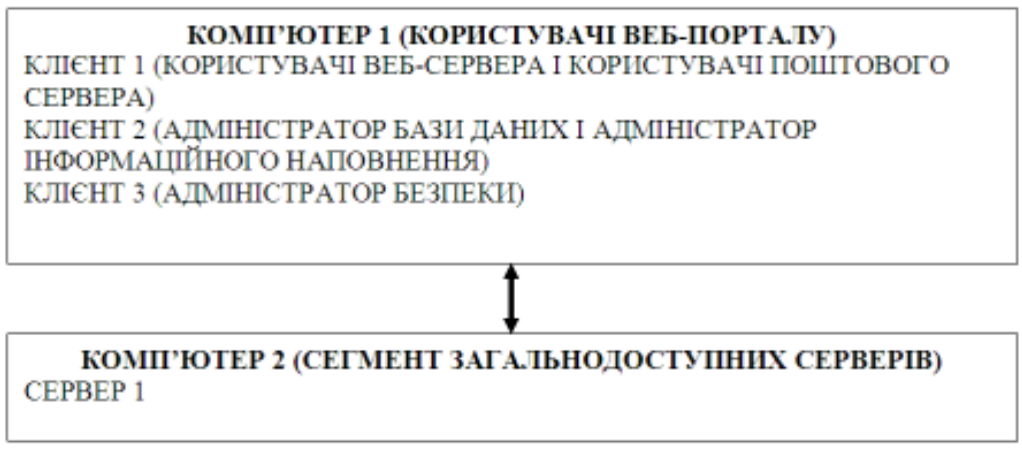
10

Логічна модель мережі веб-сайту, що використана при проведенні досліджень



11

Схема розміщення компонентів тестової програми в моделі мережі веб-сайту при проведенні експерименту на швидкість обробки вихідного трафіку



12

Загальний опис алгоритмів

- У даній роботі експеримент проводиться над моделлю мережі веб-сайту.
- При цьому для з'ясування характеристик продуктивності досліджуваних типів МЕ використовується вимір по різних прямих метриках

13

Метріки

- середній час проходження IP-пакету до сегмента ДМЗ по поштових протоколах;
- середній час проходження IP-пакету із сегмента ДМЗ у зовнішню мережу по протоколу http;
- середній час проходження IP-пакету із сегмента ДМЗ у зовнішню мережу по поштових протоколах;
- середній час проходження IP-пакету до службового сегмента;
- швидкість обробки вхідного трафіка зовнішнім МЕ являє собою кількість IP-пакетів перевірених на відповідність заданій політиці безпеки за одиницю часу (1 сек.);
- швидкість обробки вихідного трафіка зовнішнім МЕ являє собою кількість IP-пакетів перевірених на відповідність заданій політиці безпеки за одиницю часу (1 сек.);
- швидкість обробки трафіка внутрішнім МЕ являє собою кількість IP-пакетів перевірених на відповідність заданій політиці безпеки за одиницю часу (1 сек.).

14

Розроблений алгоритм ініціалізації і проведення вимірів швидкості передачі пакетів по тестовій мережі

алгоритм включає 7 основних етапів:

- синхронізація часу клієнтів і сервера.
- підготовка тестування.
- установка тестових з'єднань.
- ініціалізація процесу тестування.
- безпосередньо процес тестування - відбувається передача тестових пакетів по всіх тестових з'єднаннях з фіксацією часу оправлення на стороні сервера й часу одержання пакета на стороні клієнтів.
- обробка результатів - клієнти пересилають на сервер часи одержання тестових пакетів і загальний час проведення тестування. На сервері ці дані сортуються по групах тестових з'єднань, що імітують різні категорії користувачів, обробляються й зберігаються у файли формату XML.
- розрив тестових з'єднань.

15

Підготовка до тестування

на клієнтові починається із приймання ініціюючого повідомлення із сервера, після чого клієнт відсилає повідомлення підтверджувальне готовність ухвалювати інформацію про тестові з'єднання.

- Ця інформація ухвалюється в циклі.
- Після приймання кожного повідомлення з даними серверу посилає повідомлення підтверджувальне приймання. У випадку якщо наступне отримане повідомлення є повідомленням про завершення передачі, то клієнт посилає підтверджувальне повідомлення й переходить до наступного етапу алгоритму.

Загальна структура клієнтських і серверних алгоритмів ідентична, тому що в процесі роботи клієнти й сервер, по суті, будуть виконувати один загальний алгоритм, взаємодіяти по заздалегідь встановленому протоколу.

Відмінності між структурою клієнтських і серверних алгоритмів в основному обумовлені необхідністю сервера паралельно працювати з декількома клієнтами й обробляти результати вимірів.

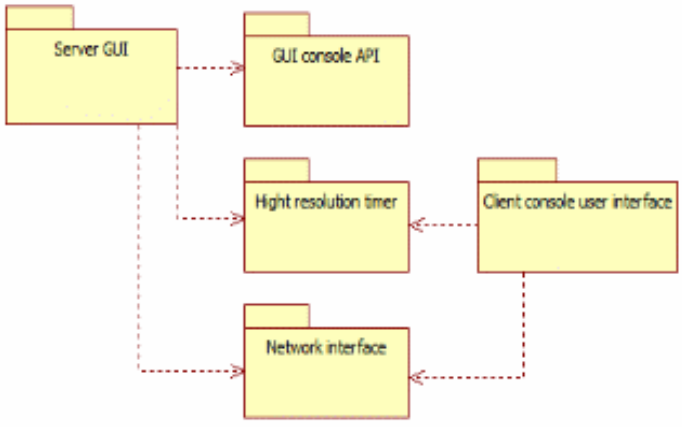
Алгоритми клієнтської й серверної частин використовують протоколи TCP/IP і UDP/IP для взаємодії один з одним

16

Діаграма розгортання програмної системи

17

Діаграма пакетів програмної системи

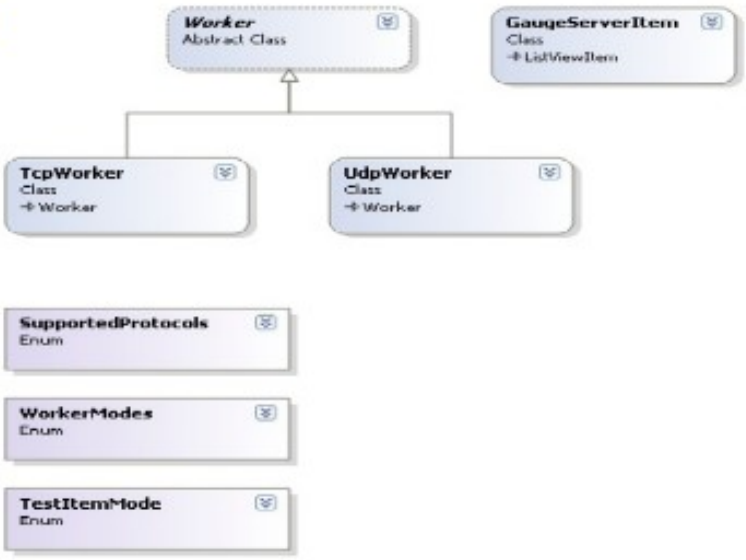


18

Діаграма прецедентів програмної системи

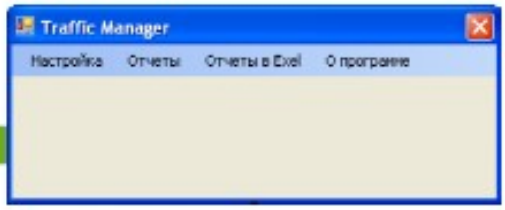
19

Діаграма класів бібліотеки Networkclient.dll

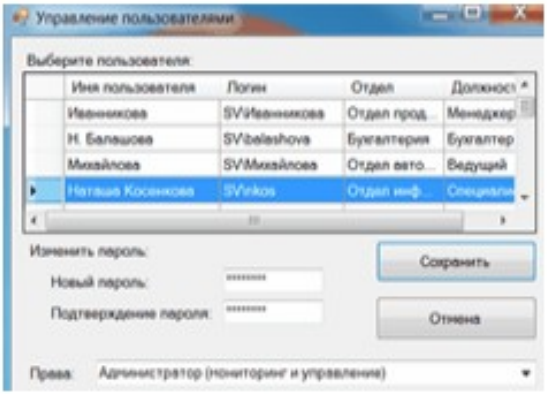


20

Головна форма додатка



Форма додатка для керування користувачами



21

Вихідні значення для тестового прикладу

	Очікувані	Отримані	Результат
Кількість елементів у списку тестових з'єднань серверного вікна «Створення нового вимірю» після видалення з'єднання з портом 5013	12	12	Пройденнй
Кількість елементів у списку тестових з'єднань серверного вікна «Створення нового вимірю» після видалення з'єднання з портом 5005	11	11	Пройденнй

22

Підсумки програмних експериментів

- загальна кількість виконаних тестових прикладів: 5;
- кількість успішна пройдених тестових прикладів: 5;
- кількість неуспішна пройдених тестових прикладів: 0;
- загальна кількість перевірених вихідних значень: 44;
- кількість вихідних значень, у яких очікуване значення не збіглося з реальним: 0.

23

Висновки

- Розроблена модель мережі веб-сайту, з якої вилучені елементи, несуттєві для вивчення проблем продуктивності.
- Обране ПЗ необхідне для проведення експериментальних досліджень у складі МЕ Kerio Winroute Firewall і ipfirewall, а також поштового сервера Mdaemon.
- Також розроблено метрики, по яких проводилися експериментальні виміри продуктивності. У результаті аналізу типових політик безпеки даних веб-сайтів розроблений план експерименту.
- Ухвалене рішення про розробку спеціалізованого ПЗ, здатного проводити тестування в багатопоточному режимі з декількома клієнтами з використанням великої кількості тестових з'єднань.
- У процесі розробки створена архітектура ПЗ задовольняючого цим умовам, протокол мережної взаємодії клієнтів і сервера, спроектовані й реалізовані відповідні модулі.
- У процесі верифікації й валідації перевірені всі вимоги ТЗ і зроблений висновок про коректність реалізації ПЗ.
- Розроблене ПЗ використане для проведення експерименту над моделлю мережі веб-сайту. Результати експерименту зафіксовані й піддані відповідній до первинної обробки.
- У результаті аналізу отриманих формул розроблений рекомендований порядок розміщення правил безпеки в списках системи МЕ в заданих умовах, що свідчить про виконання всіх завдань і досягненні мети роботи.

24

ДОДАТОК В

Апробація результатів роботи

ДОСЛІДЖЕННЯ АЛГОРИТМІЧНИХ ЗАСОБІВ ЗАБЕЗПЕЧЕННЯ МЕРЕЖЕВОЇ БЕЗПЕКИ ТА ВЕБ-САЙТІВ

Кучугура Д.В.
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 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 визирів 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) використуваних 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 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.