Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет Компьютерных Систем и Сетей

Кафедра программного обеспечения информационных технологий

Дисциплина «Метрология, стандартизация и сертификация в информационных технологиях»

|  |
| --- |
|  |

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1**

Тема: Модели оценки надёжности Программного обеспечения

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Студент группы № 053502 Кулак Андрей Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_17.10.2021  *подпись, дата* |
| Проверил: |  | Давыдовский А.Г |

Отметка\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*прописью подпись, дата*

Минск 2021

**1. Тема практического занятия №1:**

**Модели оценки надёжности программного обеспечения**

**2. Цель работы:**

Освоение студентом навыков использования наиболее широко применяемых моделей надежности программного обеспечения (ПО).

Получить навыки оценки надёжности программного обеспечения с помощью модели Коркорэна, реализовать в программной среде С# ПО для выполнения расчётов по данной модели.

Результатом выполнения практических работ должен быть отчет, содержащий расчеты оценок надежности ПО.

**3. Теоретические сведения:**

Модель содержит изменяющуюся вероятность отказов для различных источников ошибок и соответственно разную вероятность их исправления, используются такие параметры, как результат только N испытаний, в которых наблюдается Ni ошибок i-го типа;

Показатель уровня надежности R вычисляют по следующей формуле:



где N0- число безуспешных испытаний, выполненных в серии из N испытаний, k - известное число типов ошибок, Yi- вероятность появления ошибок, при Ni> 0, Yi= ai, при Ni= 0, Yi = 0.

**4. Задание для выполнения практического занятия:**

В программной среде С# реализовать расчеты показателя уровня надежности

**5. Протокол выполнения практического занятия**

**1)** Изучение методического пособия(материала) модели по теме практического задания

**2)** Программирование в программной среде С++ с использованием .NET Framework 5.0 для создания приложения.

using System;

using System.IO;

namespace MetrologyModel

{

class Program

{

static void Main(string[] args)

{

int amtUnsuccessful, amtTotal, amtErrorTypes, input;

float successChance;

float sum = 0;

string[] errorTypes = System.IO.File.ReadAllLines(@"C:\Users\andre\source\repos\MetrologyModel\ErrorTypes.txt");

double[] errorTypeChances = new double[errorTypes.Length];

int[] errorTypeOccurences = new int[errorTypes.Length];

Random random = new Random();

for (int i = 0; i < errorTypes.Length; i++)

{

errorTypeChances[i] = random.NextDouble();

}

Console.WriteLine("\n1. Run the test with semi-random data, 2. Use custom exception types and data");

while (!int.TryParse(Console.ReadLine(), out input) || (input != 1 && input != 2))

{

Console.WriteLine("Incorrect input, retry");

}

switch (input) {

case 1:

{

Console.WriteLine("\nAmount of times ran: ");

while (!int.TryParse(Console.ReadLine(), out amtTotal) || amtTotal <= 0)

{

Console.WriteLine("Incorrect input, retry");

}

Console.WriteLine("\nAmount of times ran with an exception: ");

while (!int.TryParse(Console.ReadLine(), out amtUnsuccessful) || (amtUnsuccessful <= 0 || amtUnsuccessful > amtTotal))

{

Console.WriteLine("Incorrect input, retry");

}

amtErrorTypes = errorTypes.Length;

int currentIter = 0;

for (int i = 0; i < errorTypes.Length; i++)

{

errorTypeOccurences[i] = random.Next(0, amtUnsuccessful - currentIter);

currentIter += errorTypeOccurences[i];

}

for (int i = 0; i < amtErrorTypes; i++) {

sum += (float)errorTypeChances[i] \* ((float)errorTypeOccurences[i] - 1) / (float)amtTotal;

}

successChance = ((float)amtUnsuccessful / (float)amtTotal) + sum;

Console.WriteLine("\nResulting success chance = " + successChance);

break;

}

case 2:

{

Console.WriteLine("\nAmount of custom exception types to add: ");

while (!int.TryParse(Console.ReadLine(), out input) || input <= 0)

{

Console.WriteLine("Incorrect input, retry");

}

string[] errorTypesCustom = new String[input];

for (int i = 0; i < input; i++)

{

Console.WriteLine("\nCustom exception type ", i + 1);

errorTypesCustom[i] = Console.ReadLine();

Console.WriteLine("\nCustom exception type chance (0 - 100) ", i + 1);

while (!double.TryParse(Console.ReadLine(), out errorTypeChances[i]) || (errorTypeChances[i] < 0 || errorTypeChances[i] > 100))

{

Console.WriteLine("Incorrect input, retry");

}

}

StreamWriter sw = new StreamWriter(@"C:\Users\andre\source\repos\MetrologyModel\ErrorTypes.txt");

for (int i = 0; i < input; i++)

{

sw.WriteLine(errorTypesCustom[i]);

}

sw.Close();

Console.WriteLine("\nAmount of times ran: ");

while (!int.TryParse(Console.ReadLine(), out amtTotal) || amtTotal <= 0)

{

Console.WriteLine("Incorrect input, retry");

}

Console.WriteLine("\nAmount of times ran with an exception: ");

while (!int.TryParse(Console.ReadLine(), out amtUnsuccessful) || (amtUnsuccessful <= 0 || amtUnsuccessful > amtTotal))

{

Console.WriteLine("Incorrect input, retry");

}

amtErrorTypes = errorTypes.Length;

for (int i = 0; i < amtErrorTypes; i++)

{

sum += (float)errorTypeChances[i] \* ((float)errorTypeOccurences[i] - 1) / (float)amtTotal;

}

successChance = ((float)amtUnsuccessful / (float)amtTotal) + sum;

Console.WriteLine("\nResulting success chance = " + successChance);

break;

}

}

}

}

}

**Рекомендация по использованию приложения:**

Программа поддерживает использование собственных данных, включая данные о возможных видах ошибок и их вероятностях, для этого пользователь должен выбрать соответствующюю функцию при выборе и ввести имеющиеся данные.

**6. Заключение**

Оценочные модели основываются на серии тестовых прогонов и проводятся на этапах тестирования ПC. В тестовой среде определяется вероятность отказа программы при ее выполнении или тестировании. Эти типы моделей могут применяться на этапах ЖЦ. Кроме того, результаты прогнозирующих моделей могут использоваться как входные данные для оценочной модели. Модель Коркорэна относится к статичес­ким моделям надежности ПС, так как в ней не используются па­раметры времени тестирования и учитывается только результат N испытаний, в которых выявлено N ошибок i-го типа. Модель использует изменяющиеся вероятности отказов для различных типов ошибок.

**7. Выводы**

**1)** Нами были освоены навыки использования наиболее широко применяемых моделей оценки надёжности ПО. В частности, была рассмотрена модель Коркорэна.

**2)** Реализована и применена на практике модель оценки надёжности Коркорэна, с помощью языка программирования С# было написано оконное приложение на основе фреймворка .NET 5.0