Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное   
учреждение высшего образования

«Новосибирский государственный технический университет»



Кафедра теоретической и прикладной информатики

Лабораторная работа № 8  
по дисциплине «Метрология, качество и тестирование программного обеспечения»

**Расчёт трудоёмкости разработки ПО в технологии ООП по методике Rational software**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Факультет: | ПМИ |  |  |
| Группа: | ПМИМ-01 |  |  |
| Студент: | Ничипиенко В.А. |  |  |
| Вариант: | 6 |  |  |
| Преподаватель: | Зайцев М.Г. |  |  |

Новосибирск

2021

1. **Цель работы**

Сформировать практические навыки расчёта трудоёмкости разработки ПО в технологии ООП по методике компании Rational software.

1. **Задание**
2. Спроектировать и реализовать программу в соответствии со следующей спецификацией:

* программа обеспечивает ввод исходных данных в режиме графического интерфейса для расчёта трудоёмкости разработки по методике компании Rationalsoftware;
* Осуществляет поэтапное вычисление трудоёмкости с возможность возврата к предыдущим этапам;
* Сохраняет результат и исходные данные по каждому проекту в отдельном файле;
* Позволяет просматривать файлы с расчётами трудоёмкости по предыдущим проектам.

1. **Проект программы**

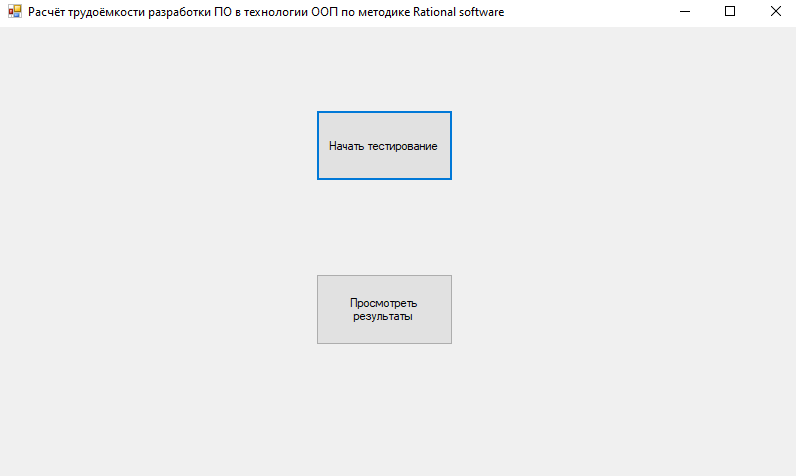
**

Рисунок 1 - Меню

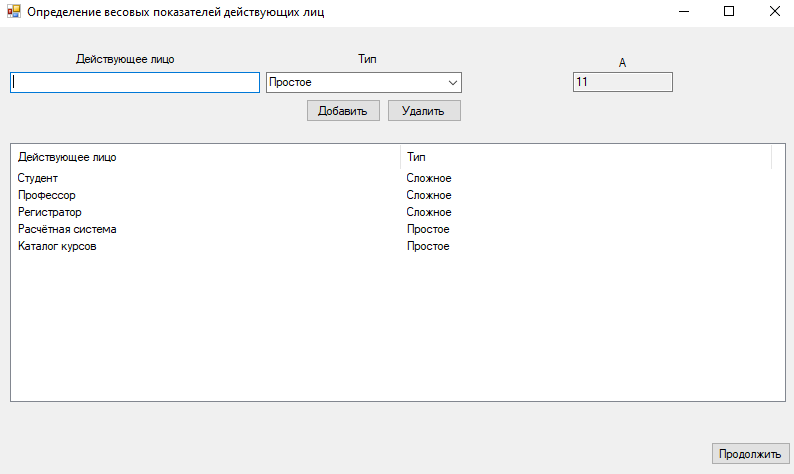
**

Рисунок 2 - Определение весовых показателей действующих лиц

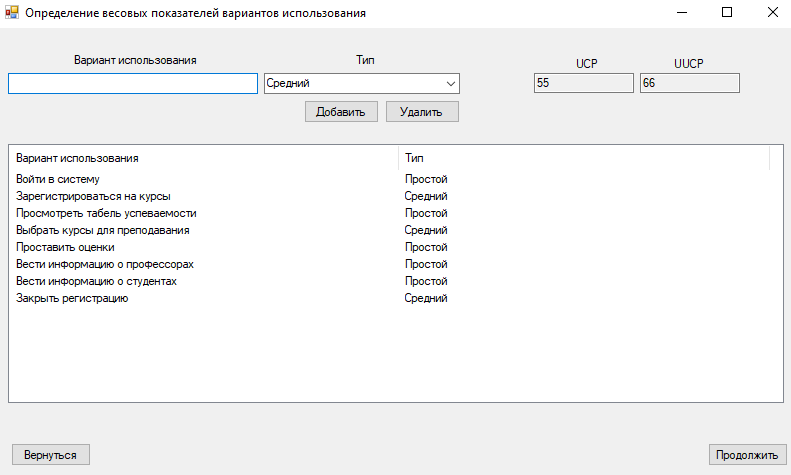
**

Рисунок 3 - Определение весовых показателей вариантов использования

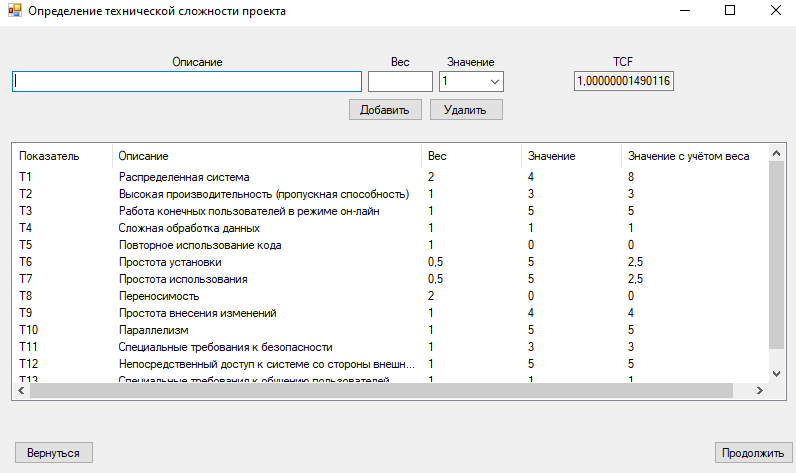
**

Рисунок 4 - Определение технической сложности проекта

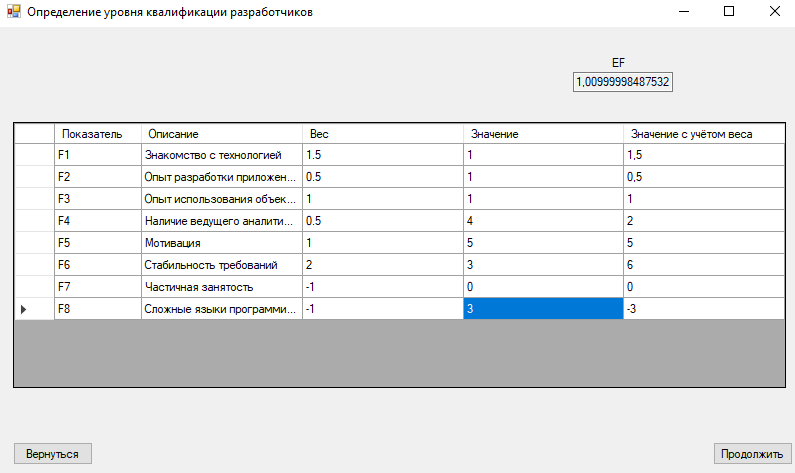
**

Рисунок 5 - Определение уровня квалификации разработчиков

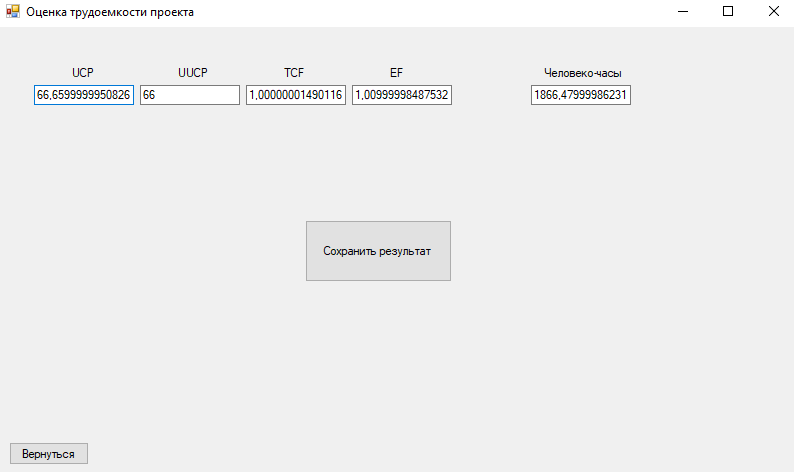
**

Рисунок 6 - Оценка трудоёмкости проекта

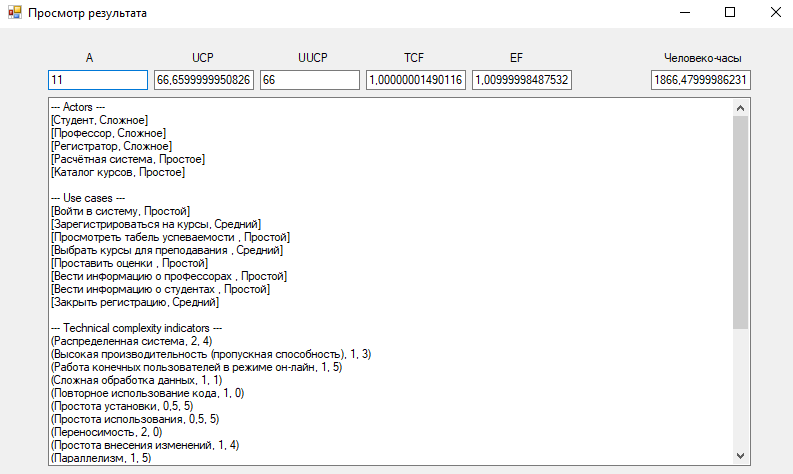
**

Рисунок 7 - Просмотр результата

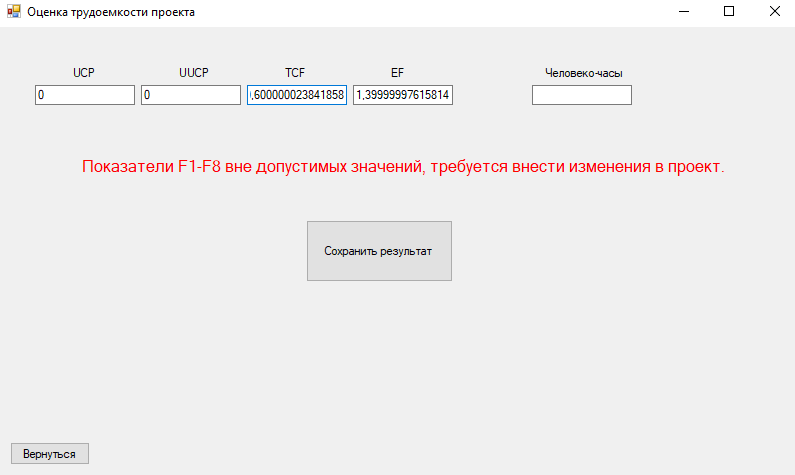
**

Рисунок 8 - Ошибка при вводе показателей F1-F8 вне допустимых значений

1. **Текст программы**

*TestData.cs*

using System;

using System.Collections.Generic;

namespace RationalSoftwareTesting

{

[Serializable]

public class TestData

{

public Dictionary<string, string> actors;

public Dictionary<string, string> useCases;

public List<(string, double, int)> technicalComplexityIndicators;

public List<(string, double, int)> environmentalIndicators;

public double A { get; set; }

public double UCP { get; set; }

public double UUCP { get; set; }

public double TCF { get; set; }

public double EF { get; set; }

public double NumberManHoursForUCP { get; set; }

public double NumberManHours { get; set; }

public TestData()

{

actors = new Dictionary<string, string>();

useCases = new Dictionary<string, string>();

technicalComplexityIndicators = new List<(string, double, int)>();

environmentalIndicators = new List<(string, double, int)>();

}

}

}

*TestController.cs*

using Newtonsoft.Json;

using System.Collections.Generic;

using System.IO;

namespace RationalSoftwareTesting

{

public class TestController

{

private TestData testData;

public TestData TestData => testData;

public TestController()

{

testData = new TestData();

}

public void CalculateA(List<double> values)

{

testData.A = 0;

foreach(var value in values)

{

testData.A += value;

}

}

public void CalculateUCP(List<double> values)

{

testData.UCP = 0;

foreach (var value in values)

{

testData.UCP += value;

}

CalculateUUCP();

}

public void CalculateUCP()

{

testData.UCP = testData.UUCP \* testData.TCF \* testData.EF;

}

public void CalculateUUCP()

{

testData.UUCP = testData.A + testData.UCP;

}

public void CalculateTCF(double value)

{

testData.TCF = 0.6f + (0.01f \* value);

}

public void CalculateEF(double value)

{

testData.EF = 1.4f + (-0.03f \* value);

}

public bool TryGetNumberManHoursForUCP(out double numberManHoursForUCP)

{

int sum = 0;

numberManHoursForUCP = testData.NumberManHoursForUCP = 0;

for (int i = 0; i <= 5; i++)

{

if (testData.environmentalIndicators[i].Item3 < 3)

{

sum++;

}

}

for (int i = 6; i < 8; i++)

{

if (testData.environmentalIndicators[i].Item3 > 3)

{

sum++;

}

}

if (sum <= 2)

{

numberManHoursForUCP = testData.NumberManHoursForUCP = 20;

return true;

}

else

{

if (sum > 2 && sum < 5)

{

numberManHoursForUCP = testData.NumberManHoursForUCP = 28;

return true;

}

else

{

return false;

}

}

}

public double GetNumberManHours(double numberManHoursForUCP)

{

testData.NumberManHours = testData.UCP \* numberManHoursForUCP;

return testData.NumberManHours;

}

public void SaveData(string fileName)

{

File.WriteAllText(fileName, JsonConvert.SerializeObject(testData));

}

}

}

*Menu.cs*

using Newtonsoft.Json;

using System;

using System.IO;

using System.Windows.Forms;

namespace RationalSoftwareTesting

{

public partial class Menu : Form

{

public Menu()

{

InitializeComponent();

}

private void buttonStartTest\_Click(object sender, EventArgs e)

{

TestController testController = new TestController();

DeterminingTheWeightIndicatorsOfActors form = new DeterminingTheWeightIndicatorsOfActors(testController);

form.Show();

}

private void buttonResults\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Test data(\*.json)|\*.json"; ;

if (openFileDialog.ShowDialog() == DialogResult.Cancel)

return;

string fileName = openFileDialog.FileName;

string jsonString = File.ReadAllText(fileName);

TestData testData = JsonConvert.DeserializeObject<TestData>(jsonString);

ResultsViewer form = new ResultsViewer(testData);

form.Show();

}

}

}

*DeterminingTheWeightIndicatorsOfActors.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace RationalSoftwareTesting

{

public partial class DeterminingTheWeightIndicatorsOfActors : Form

{

private TestController testController;

private Dictionary<string, int> typesCharacters = new Dictionary<string, int>()

{

{ "Простое", 1 },

{ "Среднее", 2 },

{ "Сложное", 3 }

};

private event Action OnFormUpdate;

public DeterminingTheWeightIndicatorsOfActors(TestController testController)

{

InitializeComponent();

this.testController = testController;

comboBox.DataSource = typesCharacters.Keys.ToList();

OnFormUpdate += UpdateListView;

OnFormUpdate += UpdateNumberA;

OnFormUpdate?.Invoke();

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(textBox.Text))

return;

testController.TestData.actors.Add(textBox.Text, comboBox.SelectedItem.ToString());

OnFormUpdate?.Invoke();

textBox.Clear();

textBox.Focus();

}

private void buttonRemove\_Click(object sender, EventArgs e)

{

if (listView.Items.Count <= 0)

return;

testController.TestData.actors.Remove(listView.SelectedItems[0].Text);

OnFormUpdate?.Invoke();

}

private void buttonContinue\_Click(object sender, EventArgs e)

{

OnFormUpdate -= UpdateListView;

OnFormUpdate -= UpdateNumberA;

DeterminingTheWeightIndicatorsOfUseCases form = new DeterminingTheWeightIndicatorsOfUseCases(testController);

form.Show();

Close();

}

private void UpdateListView()

{

listView.Items.Clear();

foreach (var actor in testController.TestData.actors)

{

ListViewItem item = new ListViewItem(actor.Key);

item.SubItems.Add(actor.Value.ToString());

listView.Items.Add(item);

}

}

private void UpdateNumberA()

{

List<double> values = new List<double>();

for (int i = 0; i < listView.Items.Count; i++)

{

values.Add(typesCharacters[listView.Items[i].SubItems[1].Text]);

}

testController.CalculateA(values);

textBoxA.Text = testController.TestData.A.ToString();

}

}

}

*DeterminingTheWeightIndicatorsOfUseCases.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace RationalSoftwareTesting

{

public partial class DeterminingTheWeightIndicatorsOfUseCases : Form

{

private TestController testController;

private Dictionary<string, int> useCases = new Dictionary<string, int>()

{

{ "Простой", 5 },

{ "Средний", 10 },

{ "Сложный", 15 }

};

private event Action OnFormUpdate;

public DeterminingTheWeightIndicatorsOfUseCases(TestController testController)

{

InitializeComponent();

this.testController = testController;

comboBox.DataSource = useCases.Keys.ToList();

OnFormUpdate += UpdateListView;

OnFormUpdate += UpdateNumberUCP;

OnFormUpdate?.Invoke();

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(textBox.Text))

return;

testController.TestData.useCases.Add(textBox.Text, comboBox.SelectedItem.ToString());

OnFormUpdate?.Invoke();

textBox.Clear();

textBox.Focus();

}

private void buttonRemove\_Click(object sender, EventArgs e)

{

if (listView.Items.Count <= 0)

return;

testController.TestData.useCases.Remove(listView.SelectedItems[0].Text);

OnFormUpdate?.Invoke();

}

private void buttonReturn\_Click(object sender, EventArgs e)

{

OnFormUpdate -= UpdateListView;

OnFormUpdate -= UpdateNumberUCP;

DeterminingTheWeightIndicatorsOfActors form = new DeterminingTheWeightIndicatorsOfActors(testController);

form.Show();

Close();

}

private void buttonContinue\_Click(object sender, EventArgs e)

{

OnFormUpdate -= UpdateListView;

OnFormUpdate -= UpdateNumberUCP;

DeterminingTheTechnicalComplexityOfTheProject form = new DeterminingTheTechnicalComplexityOfTheProject(testController);

form.Show();

Close();

}

private void UpdateListView()

{

listView.Items.Clear();

foreach (var useCase in testController.TestData.useCases)

{

ListViewItem item = new ListViewItem(useCase.Key);

item.SubItems.Add(useCase.Value.ToString());

listView.Items.Add(item);

}

}

private void UpdateNumberUCP()

{

List<double> values = new List<double>();

for (int i = 0; i < listView.Items.Count; i++)

{

values.Add(useCases[listView.Items[i].SubItems[1].Text]);

}

testController.CalculateUCP(values);

textBoxUCP.Text = testController.TestData.UCP.ToString();

textBoxUUCP.Text = testController.TestData.UUCP.ToString();

}

}

}

*DeterminingTheTechnicalComplexityOfTheProject.cs*

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Windows.Forms;

namespace RationalSoftwareTesting

{

public partial class DeterminingTheTechnicalComplexityOfTheProject : Form

{

private TestController testController;

private List<int> values = new List<int>() { 0, 1, 2, 3, 4, 5 };

private event Action OnFormUpdate;

public DeterminingTheTechnicalComplexityOfTheProject(TestController testController)

{

InitializeComponent();

this.testController = testController;

comboBox.DataSource = values;

OnFormUpdate += UpdateListView;

OnFormUpdate += UpdateNumberTCF;

OnFormUpdate?.Invoke();

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(textBoxDescription.Text) || string.IsNullOrEmpty(textBoxWeight.Text))

return;

testController.TestData.technicalComplexityIndicators.Add

((textBoxDescription.Text, double.Parse(textBoxWeight.Text, CultureInfo.InvariantCulture.NumberFormat), Convert.ToInt32(comboBox.SelectedItem)));

OnFormUpdate?.Invoke();

textBoxDescription.Clear();

textBoxWeight.Clear();

textBoxDescription.Focus();

}

private void buttonRemove\_Click(object sender, EventArgs e)

{

if (listView.Items.Count <= 0)

return;

(string, double, int) item = (listView.SelectedItems[0].SubItems[1].Text,

Convert.ToDouble(listView.SelectedItems[0].SubItems[2].Text),

Convert.ToInt32(listView.SelectedItems[0].SubItems[3].Text));

testController.TestData.technicalComplexityIndicators.Remove(item);

OnFormUpdate?.Invoke();

}

private void buttonReturn\_Click(object sender, EventArgs e)

{

OnFormUpdate -= UpdateListView;

OnFormUpdate -= UpdateNumberTCF;

DeterminingTheWeightIndicatorsOfUseCases form = new DeterminingTheWeightIndicatorsOfUseCases(testController);

form.Show();

Close();

}

private void buttonContinue\_Click(object sender, EventArgs e)

{

OnFormUpdate -= UpdateListView;

OnFormUpdate -= UpdateNumberTCF;

DeterminingTheLevelOfQualificationOfDevelopers form = new DeterminingTheLevelOfQualificationOfDevelopers(testController);

form.Show();

Close();

}

private void UpdateListView()

{

listView.Items.Clear();

for (int i = 0; i < testController.TestData.technicalComplexityIndicators.Count; i++)

{

ListViewItem item = new ListViewItem("T" + (i + 1));

item.SubItems.Add(testController.TestData.technicalComplexityIndicators[i].Item1);

item.SubItems.Add(testController.TestData.technicalComplexityIndicators[i].Item2.ToString());

item.SubItems.Add(testController.TestData.technicalComplexityIndicators[i].Item3.ToString());

item.SubItems.Add((testController.TestData.technicalComplexityIndicators[i].Item2 \*

testController.TestData.technicalComplexityIndicators[i].Item3).ToString());

listView.Items.Add(item);

}

}

private void UpdateNumberTCF()

{

double value = 0;

foreach(var elem in testController.TestData.technicalComplexityIndicators)

{

value += elem.Item2 \* elem.Item3;

}

testController.CalculateTCF(value);

textBoxTCF.Text = testController.TestData.TCF.ToString();

}

}

}

*DeterminingTheLevelOfQualificationOfDevelopers.cs*

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Windows.Forms;

namespace RationalSoftwareTesting

{

public partial class DeterminingTheLevelOfQualificationOfDevelopers : Form

{

private TestController testController;

private List<string> descriptions = new List<string>() { "description1", "description2", "description3", "description4",

"description5", "description6","description7","description8",};

private int numberFactors = 8;

private event Action OnFormUpdate;

public DeterminingTheLevelOfQualificationOfDevelopers(TestController testController)

{

InitializeComponent();

this.testController = testController;

OnFormUpdate += UpdateNumberEF;

OnFormUpdate?.Invoke();

if (testController.TestData.environmentalIndicators.Count == 0)

{

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

{

dataGridView1.Rows.Add("F" + (i + 1).ToString(), descriptions[i], 0, 0, 0);

testController.TestData.environmentalIndicators.Add((descriptions[i], 0, 0));

}

}

else

{

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

{

dataGridView1.Rows.Add("F" + (i + 1).ToString(),

testController.TestData.environmentalIndicators[i].Item1,

testController.TestData.environmentalIndicators[i].Item2,

testController.TestData.environmentalIndicators[i].Item3,

testController.TestData.environmentalIndicators[i].Item2 \* testController.TestData.environmentalIndicators[i].Item3);

}

}

}

private void buttonReturn\_Click(object sender, EventArgs e)

{

OnFormUpdate -= UpdateNumberEF;

DeterminingTheTechnicalComplexityOfTheProject form = new DeterminingTheTechnicalComplexityOfTheProject(testController);

form.Show();

Close();

}

private void buttonContinue\_Click(object sender, EventArgs e)

{

OnFormUpdate -= UpdateNumberEF;

AssessmentTheLaboriousnessOfTheProject form = new AssessmentTheLaboriousnessOfTheProject(testController);

form.Show();

Close();

}

private void UpdateNumberEF()

{

double value = 0;

foreach (var elem in testController.TestData.environmentalIndicators)

{

value += elem.Item2 \* elem.Item3;

}

testController.CalculateEF(value);

textBoxEF.Text = testController.TestData.EF.ToString();

}

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

if (dataGridView1.RowCount != numberFactors)

return;

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

{

if (Convert.ToDouble(dataGridView1[3, i].Value) < 0)

{

dataGridView1[3, i].Value = 0;

}

else

{

if (Convert.ToDouble(dataGridView1[3, i].Value) > 5)

{

dataGridView1[3, i].Value = 5;

}

}

double value1 = Convert.ToDouble(dataGridView1[2, i].Value, CultureInfo.InvariantCulture.NumberFormat);

double value2 = Convert.ToDouble(dataGridView1[3, i].Value);

dataGridView1[4, i].Value = value1 \* value2;

testController.TestData.environmentalIndicators[i] = ((dataGridView1[1, i].Value.ToString(),

Convert.ToDouble(dataGridView1[2, i].Value, CultureInfo.InvariantCulture.NumberFormat),

Convert.ToInt32(dataGridView1[3, i].Value)));

}

OnFormUpdate?.Invoke();

}

}

}

*AssessmentTheLaboriousnessOfTheProject.cs*

using System;

using System.Windows.Forms;

namespace RationalSoftwareTesting

{

public partial class AssessmentTheLaboriousnessOfTheProject : Form

{

private TestController testController;

private event Action OnFormUpdate;

public AssessmentTheLaboriousnessOfTheProject(TestController testController)

{

InitializeComponent();

this.testController = testController;

OnFormUpdate += UpdateResult;

OnFormUpdate?.Invoke();

}

private void buttonReturn\_Click(object sender, EventArgs e)

{

OnFormUpdate -= UpdateResult;

DeterminingTheLevelOfQualificationOfDevelopers form = new DeterminingTheLevelOfQualificationOfDevelopers(testController);

form.Show();

Close();

}

private void buttonSaveResult\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Test data(\*.json)|\*.json";

if (saveFileDialog.ShowDialog() == DialogResult.Cancel)

return;

string fileName = saveFileDialog.FileName;

testController.SaveData(fileName);

MessageBox.Show("Файл сохранен");

}

private void UpdateResult()

{

testController.CalculateUCP();

textBoxUCP.Text = testController.TestData.UCP.ToString();

textBoxUUCP.Text = testController.TestData.UUCP.ToString();

textBoxTCF.Text = testController.TestData.TCF.ToString();

textBoxEF.Text = testController.TestData.EF.ToString();

textBoxMH.Clear();

if (testController.TryGetNumberManHoursForUCP(out double value))

{

labelMessage.Visible = false;

value = testController.GetNumberManHours(value);

textBoxMH.Text = value.ToString();

}

else

{

labelMessage.Visible = true;

}

}

}

}

*ResultsViewer.cs*

using System;

using System.Windows.Forms;

namespace RationalSoftwareTesting

{

public partial class ResultsViewer : Form

{

public ResultsViewer(TestData testData)

{

InitializeComponent();

textBoxA.Text = testData.A.ToString();

textBoxUCP.Text = testData.UCP.ToString();

textBoxUUCP.Text = testData.UUCP.ToString();

textBoxTCF.Text = testData.TCF.ToString();

textBoxEF.Text = testData.EF.ToString();

textBoxMH.Text = testData.NumberManHours.ToString();

textBoxInitialData.Text += "--- Actors ---" + Environment.NewLine;

foreach (var elem in testData.actors)

{

textBoxInitialData.Text += elem + Environment.NewLine;

}

textBoxInitialData.Text += Environment.NewLine + "--- Use cases ---" + Environment.NewLine;

foreach (var elem in testData.useCases)

{

textBoxInitialData.Text += elem + Environment.NewLine;

}

textBoxInitialData.Text += Environment.NewLine + "--- Technical complexity indicators ---" + Environment.NewLine;

foreach (var elem in testData.technicalComplexityIndicators)

{

textBoxInitialData.Text += elem + Environment.NewLine;

}

textBoxInitialData.Text += Environment.NewLine + "--- Environmental indicators ---" + Environment.NewLine;

foreach (var elem in testData.environmentalIndicators)

{

textBoxInitialData.Text += elem + Environment.NewLine;

}

}

}

}

1. **Тестовые наборы данных**

--- Actors ---

[Студент, Сложное]

[Профессор, Сложное]

[Регистратор, Сложное]

[Расчётная система, Простое]

[Каталог курсов, Простое]

--- Use cases ---

[Войти в систему, Простой]

[Зарегистрироваться на курсы, Средний]

[Просмотреть табель успеваемости, Простой]

[Выбрать курсы для преподавания, Средний]

[Проставить оценки, Простой]

[Вести информацию о профессорах, Простой]

[Вести информацию о студентах, Простой]

[Закрыть регистрацию, Средний]

--- Technical complexity indicators ---

(Распределенная система, 2, 4)

(Высокая производительность (пропускная способность), 1, 3)

(Работа конечных пользователей в режиме он-лайн, 1, 5)

(Сложная обработка данных, 1, 1)

(Повторное использование кода, 1, 0)

(Простота установки, 0,5, 5)

(Простота использования, 0,5, 5)

(Переносимость, 2, 0)

(Простота внесения изменений, 1, 4)

(Параллелизм, 1, 5)

(Специальные требования к безопасности, 1, 3)

(Непосредственный доступ к системе со стороны внешних пользователей, 1, 5)

(Специальные требования к обучению пользователей, 1, 1)

--- Environmental indicators ---

(Знакомство с технологией, 1,5, 1)

(Опыт разработки приложений, 0,5, 1)

(Опыт использования объектно-ориентированного подхода, 1, 1)

(Наличие ведущего аналитика, 0,5, 4)

(Мотивация, 1, 5)

(Стабильность требований, 2, 3)

(Частичная занятость, -1, 0)

(Сложные языки программирования, -1, 3)

1. **Вывод по выполненной работе**

В ходе выполнения работы были сформированы практические навыки расчёта трудоёмкости разработки ПО в технологии ООП по методике компании Rational software. И было реализовано приложения для выполнения расчётов по данной методике.