УДК: 004.42

Разработка и тестирование интеллектуальной системы отбора экспертов

Паршин И.И., Зайцева Т.В., Пусная О.П., Путивцева Н.П.

Белгородский государственный национальный исследовательский университет, Белгород,

e-mail: 1061376@bsu.edu.ru

В статье рассматривается вопрос разработки и тестирования интеллектуальный систем. Были освещены основные элементы разработанной системы . Для основных элементов был представлен код обработчиков, связанных с ними событий. Было проведено тестирование разработанной системы на основе разработанной модели. В рамках тестирования был осуществлён ввод и сохранение правил, осуществлён логический вывод, а также выведено окно пояснений, отражающее ход логического вывода.

Ключевые слова: интеллектуальная система, разработка, тестирование, интеллектуальных систем, C#.

Design of an intelligent system for selection of experts

Parshin I.I, Zaitseva T.V., Pusnaya O.P., Putivzeva N.P.

Belgorod National Research Univercity, Belgorod, e-mail:

carloscondit95@gmail.com

The article deals with the development and testing of intelligent systems. The main elements of the developed system were highlighted . For the main elements, the code of handlers associated with them events was presented. The developed system was tested on the basis of the developed model. As part of the testing, rules were entered and saved, logical output was performed, and an explanation window was displayed that reflects the progress of logical output.

Keywords: intelligent system, development, testing, intelligent systems, C#.

Разрабатываемая интеллектуальная система состоит из 3 подсистем:

- подсистема работы с правилами;
- подсистема логического вывода;
- подсистема пояснений.

Подсистема работы с правилами выполняет следующие функции:

- ввод правил;
- удаление правил;
- изменение правил;
- чтение правил из CSV файла;

– сохранение правил в CSV файл.

Окно подсистемы работы с правилами представлено на рисунке 1.

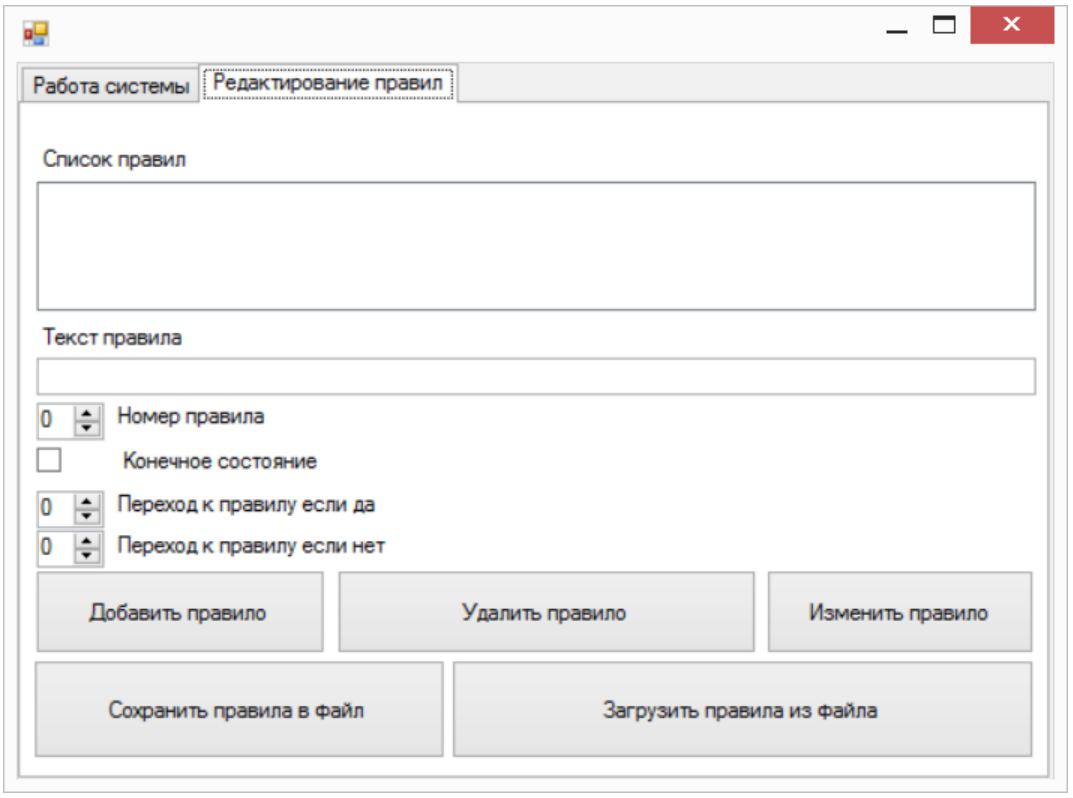

Рисунок 1 – Подсистема работы с правилами

Код обработчика события, связанного с нажатием на кнопку «Добавить правило» представлен на рисунке 2:

```
private void button3_Click(object sender, EventArgs e)
{
    int i = 0;
    for (i = 0; i < Rules.Count; i++)
    {
        if(Rules[i].ID==(int)numericUpDown1.Value)
        {
            MessageBox.Show("Правило с таким индексом уже существует");
            return;
        }
    }
    Rules.Add(new Rule() { Text = textBox1.Text, ID=(int)(numericUpDown1.Value),IsEnd=checkBox1.Checked,NextYes=
    int listcount = Rules.Count;
    i = 0;
    while (Rules[i].ID != Convert.ToInt16(numericUpDown1.Value)) i++;
    listBox1.Items.Add(Rules[i].ID.ToString()+". "+Rules[i].Text);
    int maxind = 0;
    for (i = 0; i < Rules.Count; i++)
    {
        if (Rules[i].ID > maxind) maxind = Rules[i].ID;
    }
    numericUpDown1.Value = maxind + 1;
}
```

Рисунок 2 – Код добавления правил

В данном фрагменте осуществляется проверка на наличие правил с таким же индексом, добавляется новое правило в объект Rules типа Rules После осуществляется перезаполнение списка правил в форме программы.

Код классов Rule и Rules приведён на рисунке 3.

```
public class Rule
{

    ссылок: 15
    public string Text { get; set; }
    ссылок: 27
    public int ID { get; set; }
    ссылок: 8
    public bool IsEnd { get; set; }
    ссылок: 10
    public int NextYes { get; set; }
    ссылок: 7
    public int NextNo { get; set; }
    ссылок: 0
    public Double ResultChangeYes { get; set; }
    ссылок: 0
    public Double ResultChangeNo { get; set; }

}
public List<Rule> Rules = new List<Rule>();
```

Рисунок 3 – код классов Rule и Rules

Класс Rule служит для хранения в памяти приложения информации о правилах и содеhжит следующую информацию:

– сам текст правила;
– номер правила;
– является ли правило конечным
– номер следующего правила, при ответе: «да»;
– номер следующего правила, при ответе: «да»;

Код обработчика события, связанного с нажатием на кнопку «Удалить правило» представлен на рисунке 4.

```
private void button4_Click(object sender, EventArgs e)
{
    string a = listBox1.GetItemText(listBox1.SelectedItem);
    string a1;
    int k;
    k = a.IndexOf('.');
    a1 = a.Substring(0, k);
    int deleteruleind = Convert.ToInt16(a1);
    listBox1.Items.RemoveAt(listBox1.SelectedIndex);
    int i = 0;
    while (Rules[i].ID != deleteruleind) i++;
    Rules.RemoveAt(i);
    listBox1.Items.Clear();
    for(i=0;i<Rules.Count;i++) listBox1.Items.Add(Rules[i].ID.ToString() + ". " + Rules[i].Text);

}
```

Рисунок 4 – Код удаления правил

Код обработчика события, связанного с нажатием на кнопку «Изменить правило» представлен на рисунке 5.

```
private void button5_Click(object sender, EventArgs e)
{
    string a = listBox1.GetItemText(listBox1.SelectedItem);
    string a1;
    int k;
    k = a.IndexOf('.');
    a1 = a.Substring(0, k);
    int changeruleind = Convert.ToInt16(a1);
    int i = 0;
    for (i = 0; i < Rules.Count; i++)
    {
        if (Rules[i].ID == (int)numericUpDown1.Value && Rules[i].ID!= changeruleind)
        {
            MessageBox.Show("Правило с таким индексом уже существует");
            return;
        }
    }
     i = 0;
    while (Rules[i].ID != changeruleind) i++;
    Rules[i].Text = textBox1.Text;
    Rules[i].ID = (int)(numericUpDown1.Value);
    Rules[i].IsEnd = checkBox1.Checked;
    Rules[i].NextYes = (int)(numericUpDown2.Value);
    Rules[i].NextNo = (int)(numericUpDown3.Value);
    listBox1.Items.Clear();
    for (i = 0; i < Rules.Count; i++) listBox1.Items.Add(Rules[i].ID.ToString() + ". " + Rules[i].Text);
}
```

Рисунок 5 – Код изменения правил

Код обработчика события, связанного с нажатием на кнопку «Сохранить правила в файл» представлен на рисунке 6.

```
private void button1_Click(object sender, EventArgs e)
{
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter= "CSV files (*.CSV)|*.CSV";
    try
    {
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            string file = sfd.FileName;
            using (var sw = new StreamWriter(@file, false, Encoding.Default))
            {
                sw.WriteLine("Text;ID;IsEnd;NextYes;NextNo;ResultChangeYes;ResultChangeNo");
                for (int i = 0; i < Rules.Count; i++)
                {
                    sw.WriteLine(Rules[i].Text.ToString() + ";" + Rules[i].ID.ToString() + ";" + Rules[i].IsEnd.ToString() + ";"
                }
            }
        }
    }
    catch (Exception)
    {
        // Не удалось сохранить файл
    }
}
```

Рисунок 6 – Код сохранения правил

Код обработчика события, связанного с нажатием на кнопку «Загрузить правила из файла» представлен на рисунке 7.

```
private void button2_Click_1(object sender, EventArgs e)
{
    DialogResult dialogResult = MessageBox.Show("При открытии файла текущие данные будут утеряны, желаете продолжить?", "Подтверждение", MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes)
    {
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.Filter = "CSV files (*.CSV)|*.CSV";
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            string file = ofd.FileName;
            Rules.Clear();
            using (StreamReader sr = new StreamReader(file, Encoding.Default))
            {
                string[] headers = sr.ReadLine().Split(';');

                while (!sr.EndOfStream)
                {
                    string[] rows = sr.ReadLine().Split(';');
                    Rules.Add(new Rule() { Text = rows[0].ToString(), ID = Convert.ToInt16(rows[1]), IsEnd = Convert.ToBoolean(rows[2]), NextYes = Convert.ToInt16(rows[3]),
                }

            }
            listBox1.Items.Clear();
            for (int i = 0; i < Rules.Count; i++) listBox1.Items.Add(Rules[i].ID.ToString() + ". " + Rules[i].Text);
        }

    }
    else if (dialogResult == DialogResult.No)
    {
        return;
    }
}
```

Рисунок 7 – Код открытия правил

Подсистема логического вывода осуществляет следующие функции:

– последовательное задание вопросов;

– ответ на вопросы правил;

– вывод итоговых результатов;

– вызов подсистемы пояснений.

Экранная формы логического вывода приведена на рисунке 8.

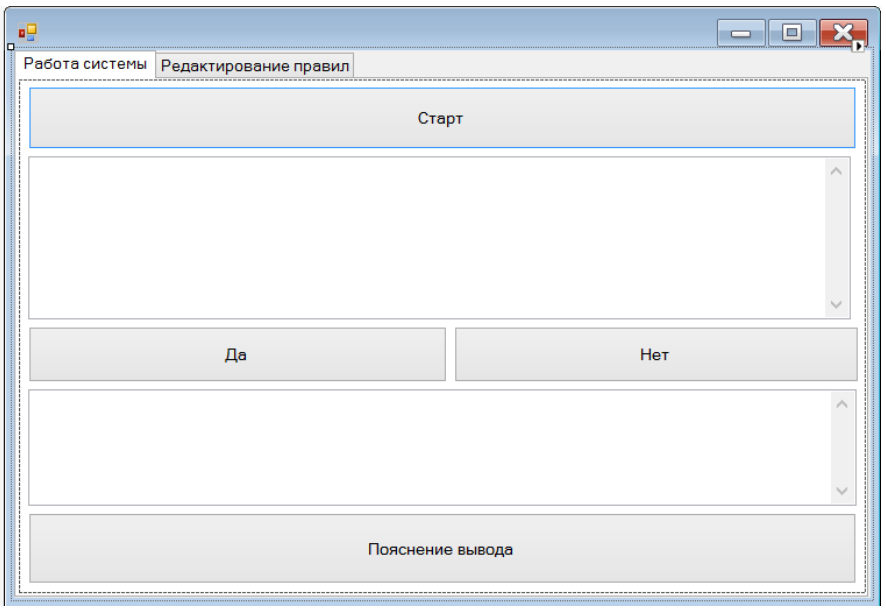

Рисунок 8 – Форма логического вывода

Код обработчика события, связанного с нажатием на кнопку «Старт» представлен на рисунке 9.

```
private void button6_Click(object sender, EventArgs e)
{
    if (Rules.Count == 0)
    {
        MessageBox.Show("Не введено ни одного правила");
        return;
    }
    int i = 0;
    bool EndExists=false;
    for (i=0;i<Rules.Count;i++)
    {
        if (Rules[i].IsEnd == true) EndExists = true;
    }
    if (EndExists==false)
    {
        MessageBox.Show("Нет ни одного конечного состояния");
        return;
    }
    PosledList.Clear();
    CurrentRuleId = 1;
    itogind = 0;
    RuleCount = 0;
    button7.Visible = true;
    button7.Enabled = true;
    button8.Visible = true;
    button8.Enabled = true;
    textBox2.Visible = true;
    i = 0;
    while (Rules[i].ID != 1) i++;
    textBox2.Text = Rules[i].Text;
    textBox3.Clear();
    button9.Enabled = false;
    button9.Visible = false;
    for(i=0;i<Rules.Count;i++)
    {
        if (Rules[i].IsEnd != true)
        {
            RuleCount += 1;
        }
    }
}
```

Рисунок 9 – Код старта

Код обработчика события, связанного с нажатием на кнопку «Да» представлен на рисунке 10.

```
private void button7_Click(object sender, EventArgs e)
{
    int i = 0;
    while (Rules[i].ID != CurrentRuleId) i++;
    for(int j=0;j<Rules.Count;j++)
    {
        if(Rules[j].ID == Rules[i].NextYes)
        {
            break;
        }
        else
        {
            if(j == Rules.Count-1)
            {
                MessageBox.Show("Для правила " + CurrentRuleId + " не существует следующего при ответе Да");
                return;

            }
        }
    }
    int yescount = 0;
    int nocount = 0;
    for (int j = 0; j < PosledList.Count; j++)
    {
        if (PosledList[j].Answer == 0) nocount++;
        else yescount++;
    }
    yescount++;
    itogind = (yescount + 0.0) / (yescount+nocount);
    PosledList.Add(new Posled { RulesListInd = i, RuleInd = CurrentRuleId, Text=Rules[i].Text,Answer=1,Currentitogind=itogind });
    CurrentRuleId = Rules[i].NextYes;
    i = 0;
    while (Rules[i].ID != CurrentRuleId) i++;
    textBox2.Text = Rules[i].Text;
}
```

Рисунок 10 – Код ответа «Да»

Ключевую роль в данном обработчике играет объект PosledList, в котором хранится информация о пройдённых правилах. Код данного объекта представлен на рисунке 11.

```
public List<Rule> Rules = new List<Rule>();
ссылок: 4
public struct  Posled
{
    public int RulesListInd;
    public int RuleInd;
    public string Text;
    public int Answer;
    public double Currentitogind;
}
public  List<Posled> PosledList = new  List<Posled>();
```

Рисунок 11 – Код объекта PosledList

Код обработчика события, связанного с нажатием на кнопку «Нет» аналогичен таковому для положительного ответа, за исключением внесения другого значения в PosledList.

Подсистема обрабатывает правила за счёт обработки события изменения верхнего текстового окна. Если текс в окне был заменён и текущее правило является конечным, то кнопки ответов блокируются, в нижнем окне выводиться итоговый результат, а также становиться доступным вызов подсистемы пояснения.

Система пояснения выводит на экран последовательность прохождения правил. Код данной подсистемы представлен на рисунке 12.

```
private void Form2_Shown(object sender, EventArgs e)
{
    Form1 frm1 = this.Owner as Form1;
    int i=0;
    for ( i = 0; i < frm1.PosledList.Count;i++)
    {
        textBox1.Text += "Правило " + frm1.PosledList[i].RuleInd + ": " + frm1.PosledList[i].Text + Environment.NewLine;
        textBox1.Text += "Ваш ответ - ";
        if (frm1.PosledList[i].Answer == 0) textBox1.Text += "Нет. ";
        else textBox1.Text += "Да: КИК + "+"1/N. ";
        textBox1.Text += "Текущий КИК = " + frm1.PosledList[i].Currentitogind+". ";
        if (frm1.PosledList[i].Answer == 0) textBox1.Text +="Следующее правило - " +frm1.Rules[frm1.PosledList[i].RulesListInd].I
        else textBox1.Text += "Следующее правило - "+ frm1.Rules[frm1.PosledList[i].RulesListInd].NextYes.ToString()+".";
        textBox1.Text += Environment.NewLine + Environment.NewLine;
    }
    int lastind = frm1.Rules[frm1.PosledList[i-1].RulesListInd].NextYes;
    textBox1.Text += "Итог :";
    for (i = 0; i < frm1.Rules.Count; i++)
    {
        if(frm1.Rules[i].ID==lastind) textBox1.Text += frm1.Rules[i].Text+". ";
    }
    textBox1.Text += "Конечный КИК: " + frm1.itogind.ToString();

}
```

Рисунок 12 – Код подсистемы пояснения

Для тестирования интеллектуальной системы в был осуществлён ввод правил из разработанного дерева решений. Результат данного ввода приведён на рисунке 13.

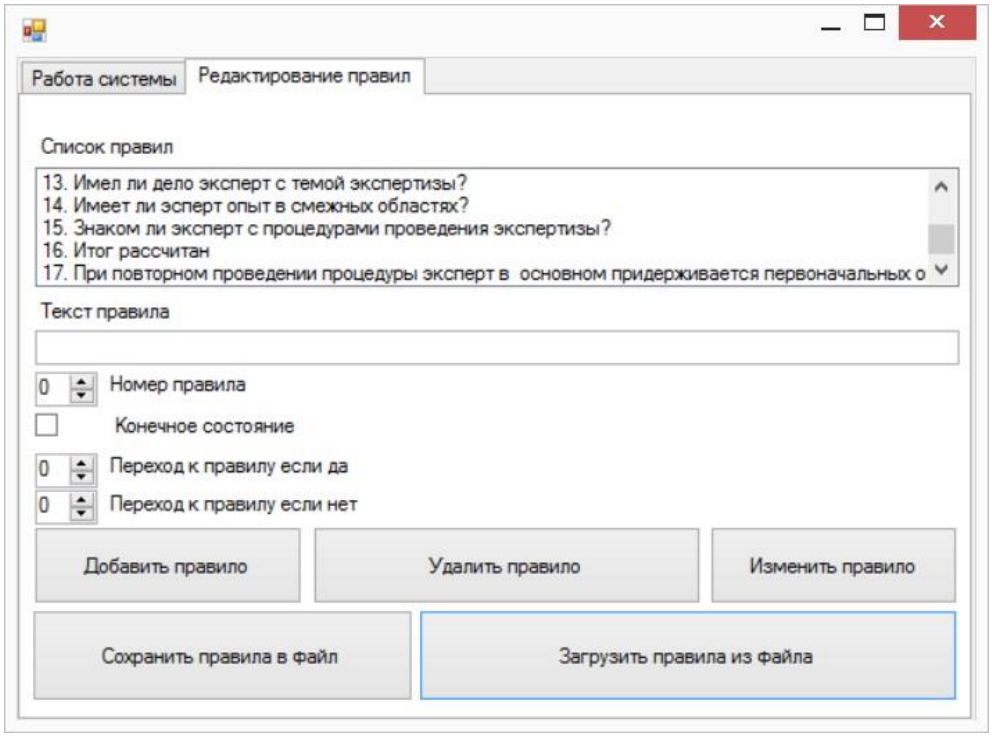

Рисунок 13 – Результат ввода правил

Во избежание потери данных было осуществлено сохранение правил в CSV файл. Данный файл представлен на рисунке 14.

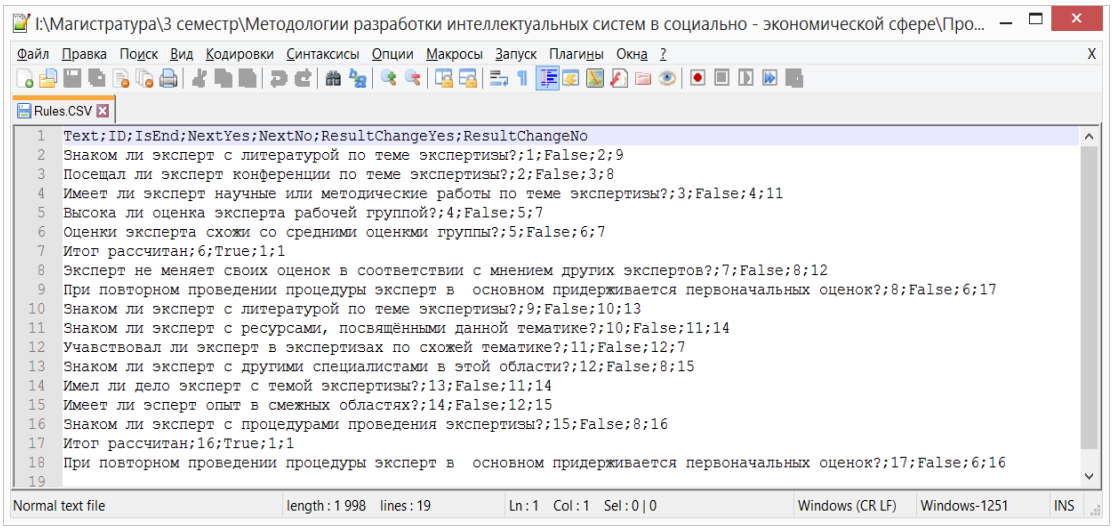

Рисунок 14. – Файл сохранения

После этого осуществлён логический вывод в соответствующей подсистеме, результат которого приведён на рисунке 15.

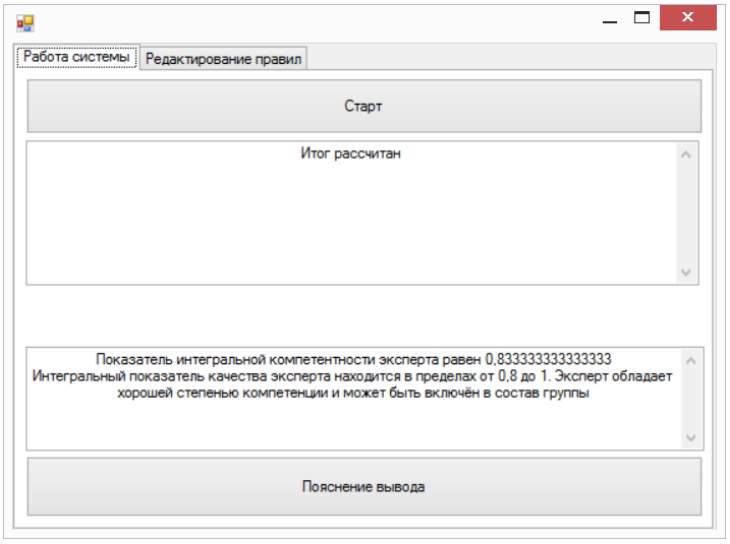

Рисунок 15 – Логический вывод

После этого был осуществлён вызов подсистемы пояснения, результат которого представлен на рисунке 16.

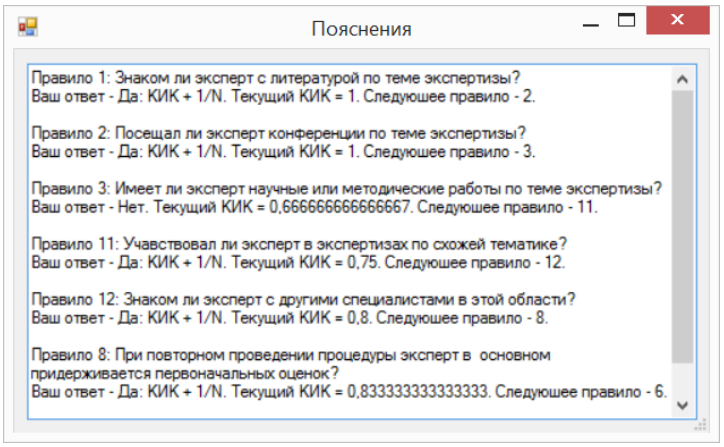

Рисунок 16 – Подсистема пояснения

Во время тестирования все функции системы осуществлялись без сбоев, а вывод и пояснения были корректными, что позволяет признать систему работоспособной.

Список литературы:

1. Visual Studio [Электронный ресурс] / visualstudio.microsoft.com – Режим доступа: https://visualstudio.microsoft.com/ru/?rr=https%3A%2F%2Fyandex.ru%2F, свободный. (дата обращения: 7.12.2019)
2. Полное руководство по языку программирования C# 8.0 и платформе .NET Core 3 [Электронный ресурс] / metanit.com – Режим доступа: https://metanit.com/sharp/tutorial/, свободный. (дата обращения: 12.12.2019)
3. Разработка интеллектуальных систем [Электронный ресурс] / studme.org – Режим доступа: https://studme.org/200942/informatika/razrabotka_intellektualnyh_sistem, свободный. (дата обращения: 10.12.2019)