МЕТОДИКА СОЗДАНИЯ РАСЧЕТНЫХ МОДУЛЕЙ ДЛЯ ПЛАТФОРМЫ OpenPSI

Варламов Д.В.

Платформа OpenPSI[1] предназначена для автоматизации работы при решении ряда задач. Платформу необходимо рассматривать с двух точек зрения: с точки зрения пользователя и с точки зрения разработчика.

С точки зрения пользователя платформа предоставляет следующий функционал:

1. Автоматизированный расчет последствий различных техногенных аварий в соответствии с утвержденными методиками;
2. Различные инструменты для анализа риска (деревья событий, расчет пожарного риска);
3. Взаимодействие с другими пользователями платформы в виде форумного общения, базы справочной информации, пополняемой самими пользователями, совместного использования и пополнения примеров готовых документов.

С точки зрения разработчика (специалиста, реализующего алгоритмы автоматизации различного рода задач) платформа предоставляет следующий функционал:

1. Фреймворк для разработки расчетных модулей с возможностью распределенных вычислений;
2. Фреймворк для объединения различных модулей в логические блоки (бизнес-процессы[2]) для решения комплексных задач;
3. Фреймворк для генерации конечных документов в форматах PDF, RTF готовых к печати;
4. Систему аутентификации и распределения доступа;
5. Доступ к разработанным модулям через веб-сервисы[3];
6. Доступ к разработанным модулям через веб-интерфейс (доступ через браузер).

Использование данной платформы удобно, в первую очередь, для конечного пользователя по следующим причинам:

1. Доступ к программным реализациям расчетных модулей осуществляется через Web, т.е. у пользователя нет необходимости скачивать и устанавливать какое либо программное обеспечение;
2. Для работы с системой необходим только браузер и подключение к сети интернет;
3. Пользователь получает результаты в удобном виде. Если модуль не требует сложных вычислений, то результат пользователь получает сразу же. Если модуль требует каких либо сложных и длительных по времени вычислений, то пользователь ожидает получения уведомление о завершении расчета, после чего может получить результаты;
4. Обновление версий происходит для пользователя незаметно;
5. Все пользователи работают с одинаковыми моделями, следовательно, возможно сопоставление результатов.

Рассмотрим процесс создания расчетного модуля с точки зрения разработчика на примере задачи расчета последствий от взрыва расширяющихся паров вскипающей жидкости (BLEVE или «огненный шар»).

РАЗРАБОТКА РАСЧЕТНОГО МОДУЛЯ

На первом этапе выбирается методика, по которой будет проводиться расчет. Методика расчета последствий для данной аварии описана в ГОСТ Р 12.3.047-98.

Расчетный модуль состоит из трех блоков: дескриптор исходных данных, дескриптор результатов расчетов (и промежуточных данных, требуемых в конечном документе) и функция (метод), осуществляющая расчет.

Требуемые исходные данные:

1. Масса вещества, участвующая в аварийном сценарии (кг);
2. Тип вещества (пропан, бутан и т.д.);

3. Расстояние от поверхности, где находится человек, до нижней границы емкости (м);

Для описания реального резервуара не всегда удобно использовать массу вещества, участвующего в аварийном сценарии. Поэтому используются дополнительные параметры для описания характеристик резервуара:

1. Объем ($м^3$);
2. Заполнение резервуара веществом (%).

Из методики берутся данные, получаемые в результате расчета, а также промежуточные данные:

1. Эффективный диаметр "огненного шара" (м);
2. Высота центра шара (м);
3. Время существования огненного шара (с);
4. Среднеповерхностная интенсивность теплового излучения пламени ($кВт/м^2$);
5. Масса вещества, участвующая в аварийном сценарии (кг);
6. Набор контрольных точек от центра «огненного шара» до точки минимального безопасного расстояния. Каждая точка характеризуется следующими показателями:
    a. Расстояние от центра «огненного шара» (м);
    b. Значение пробит-функции;
    c. Вероятность гибели человека;
    d. Коэффициент пропускания атмосферы;
    e. Угловой коэффициент облученности;
    f. Интенсивность теплового излучения огненного шара ($кВт/м^2$).

На следующем этапе создается программная реализация дескрипторов исходных данных и данных, получаемых в результате расчета.

Расчетные модули, как и все компоненты платформы, пишутся на языке Java с использованием платформы OpenPSI. В качестве среды разработки используется Eclipse[4].

Создается новый проект в Eclipse с названием *Bleve*. В список библиотек, используемых в проекте, добавляются следующие файлы:

1. FxWebServicePlatform.jar – библиотека для разработки расчетных модулей;
2. xstream-1.3.1.jar – библиотека для сериализации данных[5];
3. commons-math-2.2.jar – библиотека для работы с различными математическими функциями[6].

Создается новый пакет (package) с именем *org.intd.methods.bleve*. В пакет добавляется новый класс *ItBleveDataDescIn*. В этом классе определяются исходные данные, требуемые для выполнения расчета. Исходный текст класса приведен в листинге 1.

Листинг 1 - Исходные данные, требуемые для выполнения расчета.

```
package org.intd.methods.bleve;
import com.thoughtworks.xstream.annotations.XStreamAlias;
@XStreamAlias("data")
public class ItBleveDataDescIn
{
 @XStreamAlias("deltaHeight")
 public double  iDeltaHeight = 0.0;//Расстояние до нижней границы
 @XStreamAlias("substanceMass")
 public double  iSubstanceMass = 0.0;//Масса вещества (кг)
 @XStreamAlias("substanceID")
```

```
  public long    iSubstanceID = 0;//Идентификатор вещества
  @XStreamAlias("containerVolume")
  public double  iContainerVolume = 0.0;//Объем емкости (м^3)
  @XStreamAlias("containerFill")
  public double  iContainerFill = 0.0;//Заполненность емкости (%)
  @XStreamAlias("isUseMass")
  public boolean  iIsUseMass = true;//true - использовать указанную массу
  @XStreamAlias("controlPointsCount")
  public int    iControlPointsCount = 10;  //Кол-во контрольных
}
```

Как видно из листинга 1, дескриптор данных представляет собой класс без методов, содержащий только публичные поля, которые определяют исходные данные, и аннотации XStream[7] для сериализации. Аннотации используются для преобразования данных, описываемых классом к XML представлению (процесс сериализации), и для обратного преобразования от XML к классу (десериализация). Сериализованные данные используются для генерации конечных документов (PDF, RTF) и доступа к расчетным модулям через веб-сервисы. Аннотации задаются через знак @ и имя, следующее за ним, например:

```
@XStreamAlias("substanceMass")
public double   iSubstanceMass = 0.0;//Масса вещества
```

В данном примере задана аннотация XStream - *@XStreamAlias("substanceMass").* Данная запись означает, что при сериализации поле *iSubstanceMass* класса получит имя *substanceMass* и в XML виде будет иметь вид:

…
<*substanceMass*>значение</*substanceMass*>
…

Создается еще один класс с именем *ItBleveDataDescOut.* В этом классе определяются данные, получаемые в результате расчета. Исходный текст класса приведен в листинге 2.

Листинг 2 - Данные получаемые в результате расчета.

```
package org.intd.methods.bleve;
import com.thoughtworks.xstream.annotations.XStreamAlias;
@XStreamAlias("data")
public class ItBleveDataDescOut
{
  @XStreamAlias("diameter")
  public double ds = 0.0;//Эффективный диаметр "огненного шара" (м)
  @XStreamAlias("height")
  public double h = 0.0;//Высота центра шара (м)
  @XStreamAlias("time")
  public double t = 0.0;//Время существования огненного шара (с)
  @XStreamAlias("e")
  public double e = 450.0;//Среднеповерхностная интенсивность теплового излучения
  @XStreamAlias("m")
  public double m = 0.0;//Масса вещества (кг)
  @XStreamAlias("probit")
  public double[] probit = new double[0];//Значение пробит функции
  @XStreamAlias("p")
  public double[] p = new double[0];//Вероятность гибели человека
```

```
 @XStreamAlias("tau")
 public double[] tau = new double[0];//Коэффициент пропускания атмосферы
 @XStreamAlias("fq")
 public double[] fq = new double[0];//Угловой коэффициент облученности
 @XStreamAlias("q")
 public double[] q = new double[0];//Интенсивность теплового излучения огненного шара
 @XStreamAlias("r")
 public double[] r = new double[0];//Расстояние (м)
}
```

После того, как определены дескрипторы входных и выходных данных, создается класс с именем *ItMethodBleve*, реализующий методику расчета последствий «огненного шара». Исходный текст класса приведен в листинге 3.

Листинг 3 – Реализация методики расчета последствий «огненного шара».

```
package org.intd.methods.bleve;
import org.apache.commons.math.MathException;
import org.intd.methods.ItMethodCalcInterface;
import org.intd.openpsi.ItOpenPsiInstance;
import org.intd.substance.ItSubstanceAttrDescDB;
import org.intd.substance.ItSubstanceAttrValueDescDB;
import com.fox.db.FxConnectionDB;
import com.fox.exceptions.FxException;
import com.fox.sp.sessions.FxSPSession;
public class ItMethodBleve implements ItMethodCalcInterface
{
 private class QDesc
 {
  public double  probit  = 0.0;  //Значение пробит функции
  public double  p       = 0.0;  //Вероятность гибели человека
  public double  ds      = 0.0;  //Эффективный диаметр "огненного шара" (м)
  public double  h       = 0.0;  //Высота центра шара (м)
  public double  t       = 0.0;  //Время существования огненного шара (с)
  public double  tau     = 0.0;  //Коэффициент пропускания атмосферы
  public double  fq      = 0.0;
  public double  q       = 0.0;  //Интенсивность теплового излучения огненного шара
 }

 public Object Calc ( final Object aInDataDesc, final ItOpenPsiInstance aOpenPSI,
                      final FxSPSession aSession, final FxConnectionDB aConnection
                     ) throws FxException
 {
  //Получаем дескриптор входных данных
  ItBleveDataDescIn inDesc   = (ItBleveDataDescIn)aInDataDesc;
  ItBleveDataDescOut outDesc = new ItBleveDataDescOut();

  //Вычисляем массу вещества
  double mass = 0.0;

  if (inDesc.iIsUseMass == true)
  {
   //Берем указанную массу вещества
```

```
 mass = inDesc.iSubstanceMass;
 outDesc.e = 450.0;
} else
{
 //Рассчитываем массу вещества
 …
 Проверка значений входных данных (исходный код не приводится)
 …

 //Получаем плотность жидкой фазы из базы данных
 ItSubstanceAttrValueDescDB denstyValueDescDB = new
        ItSubstanceAttrValueDescDB(aConnection);
 denstyValueDescDB.Select(inDesc.iSubstanceID,
        ItSubstanceAttrDescDB.Type.DENSITY_OF_THE_LIQUID_PHASE);

 //Среднеповерхностная интенсивность теплового излучения пламени
 ItSubstanceAttrValueDescDB flameHeatValueDescDB = new
        ItSubstanceAttrValueDescDB(aConnection);
 flameHeatValueDescDB.Select(inDesc.iSubstanceID,
        ItSubstanceAttrDescDB.Type.FLAME_HEAT_EMMISSION);
 outDesc.e = flameHeatValueDescDB.iValue;

 //Вычисляем массу исходя из указанных параметров
 mass = inDesc.iContainerVolume * (inDesc.iContainerFill/100.0) *
      denstyValueDescDB.iValue;
}
outDesc.m = mass;

…
(исходный код не приводится)
Проверяем значение массы
Проверяем количество контрольных точек
Проверяем расстояние до нижней границы емкости
…

//Инициализируем результат
outDesc.probit = new double[inDesc.iControlPointsCount];
outDesc.p      = new double[inDesc.iControlPointsCount];
outDesc.fq     = new double[inDesc.iControlPointsCount];
outDesc.q      = new double[inDesc.iControlPointsCount];
outDesc.r      = new double[inDesc.iControlPointsCount];
outDesc.tau    = new double[inDesc.iControlPointsCount];

//Вычисляем минимальное безопасное расстояние для незащищенного человека
double  minR     = 0.0;
double  maxR     = 10000.0;
double  currentR = 0.0;
double  needQ    = 0.8;//Без негативных последствий (1.4 КВт)
double  eps      = Double.MAX_VALUE;

while (eps > 0.01)
{
```

```
    currentR = minR + (maxR - minR)/2.0;
    QDesc  qDesc  = CalcQ(mass, outDesc.e, currentR, inDesc.iDeltaHeight);
    double  d    = needQ - qDesc.q;
    if (d > 0.0) maxR = currentR;
    else minR = currentR;
    eps = Math.abs(d)/needQ;
   }

   //Вычисляем значения в каждой контрольной точке
   minR = 0.0;
   maxR = currentR;
   double step = (maxR - minR)/(double)inDesc.iControlPointsCount;
   for (int i = 0; i < inDesc.iControlPointsCount; i++)
   {
    currentR = minR + step*(double)i;
    QDesc  qDesc  = CalcQ(mass, 450.0, currentR, inDesc.iDeltaHeight);
    if (i == 0)
    {
     outDesc.ds  = qDesc.ds;
     outDesc.h  = qDesc.h;
     outDesc.t  = qDesc.t;
    }

    outDesc.probit[i] = qDesc.probit;
    outDesc.p[i]      = qDesc.p;
    outDesc.fq[i]     = qDesc.fq;
    outDesc.q[i]      = qDesc.q;
    outDesc.r[i]      = currentR;
    outDesc.tau[i]    = qDesc.tau;
   }

   //Возвращаем результат
   return outDesc;
  }
  /**
   * Метод вычисляет интенсивность теплового излучения огненного шара (кВт/м^2)
   * @param aMass - масса вещества (кг)
   * @param aEf - среднеповерхностная плотность теплового излучения пламени (кВт/м^2)
   * @param aR - расстояние до точки наблюдения (м)
   * @return Дескриптор QDesc
   */
  private QDesc CalcQ ( double aMass, double aEf, double aR, double aDeltaHeight )
  {
   QDesc res = new QDesc();
   //Эффективный диаметр "огненного шара" (м)
   res.ds = 5.33*Math.pow(aMass, 0.327);

   //Высота центра шара (м)
   res.h = res.ds/2.0 + aDeltaHeight;

   //Время существования огненного шара (с)
   res.t = 0.92*Math.pow(aMass, 0.303);
```

```
    //Коэффициент пропускания атмосферы
    double e = -0.0007*(Math.pow(aR*aR + res.h*res.h, 0.5) - res.ds/2.0);
    res.tau = Math.exp(e);

    res.fq = (res.h/res.ds + 0.5)/(4.0*Math.pow(Math.pow(res.h/res.ds + 0.5, 2.0) +
Math.pow(aR/res.ds, 2.0), 1.5));

    //Интенсивность теплового излучения огненного шара (кВт/м^2)
    res.q = aEf*res.fq*res.tau;

    //Значение пробит функции
    res.probit  = -12.8 + 2.56*Math.log(res.t*Math.pow(res.q, 4.0/3.0));

    //Вероятность гибели
    res.p = (1.0 + math.special.Erf.erf(res.probit/Math.sqrt(2.0)))/2.0;

    return res;
  }
}
```

Класс *ItMethodBleve* реализует интерфейс *ItMethodCalcInterface*, а именно метод *Calc*. Данный Интерфейс является общим для всех вычислительных модулей платформы. Метод *Calc* на вход получает следующие параметры:

1. *Object aInDataDesc* – дескриптор исходных данных;
2. ItOpenPsiInstance aOpenPSI – дескриптор данных текущего проекта;
3. FxSPSession aSession – дескриптор сессии;
4. FxConnectionDB aConnection – соединение с базой данных.

На выходе метод дает дескриптор данных, получаемых в результате расчета. В приведенном примере используются только дескриптор исходных данных и соединение с базой данных.

Первым действием при вызове метода *Calc* происходит приведение *aInDataDesc* к дескриптору *ItBleveDataDescIn*.

```
ItBleveDataDescIn inDesc   = (ItBleveDataDescIn)aInDataDesc;
ItBleveDataDescOut outDesc = new ItBleveDataDescOut();
```

При этом считается, что при вызове расчетного модуля объект *aInDataDesc* является экземпляром класса исходных данных именно вызываемого модуля. Такое поведение обеспечивает внутренний механизм платформы.

Следующий момент – получение справочной информации о свойствах веществ из базы данных, а именно информация о плотности жидкой фазы и среднеповерхностной интенсивности теплового излучения пламени. Для получения справочной информации предназначен специальный класс *ItSubstanceAttrValueDescDB*. Например, чтобы получить плотность жидкой фазы пропана необходимо создать экземпляр класса *ItSubstanceAttrValueDescDB* с указанием подключения *aConnection* к базе данных:

```
ItSubstanceAttrValueDescDB denstyValueDescDB = new
ItSubstanceAttrValueDescDB(aConnection);
```

А затем вызвать метод *Select* у экземпляра *denstyValueDescDB* с указанием идентификатора вещества (*inDesc.iSubstanceID* – целое положительное число) и требуемого свойства (*DENSITY_OF_THE_LIQUID_PHASE* – плотность жидкой фазы):

```
denstyValueDescDB.Select(inDesc.iSubstanceID,
ItSubstanceAttrDescDB.Type.DENSITY_OF_THE_LIQUID_PHASE);
```

В случае если вещество с указанным идентификатором будет найдено и в справочнике имеется значение требуемого свойства, то значение свойства будет записано в поле *iValue* класса *denstyValueDescDB*.

ИСПОЛЬЗОВАНИЕ РАСЧЕТНОГО МОДУЛЯ

Ниже приведен пример использования расчетного модуля. На первом шаге пользователь входит в систему под своим логином и паролем. После этого пользователь переходит в меню «Расчетные модули->Огненный шар» (Рис.1.)

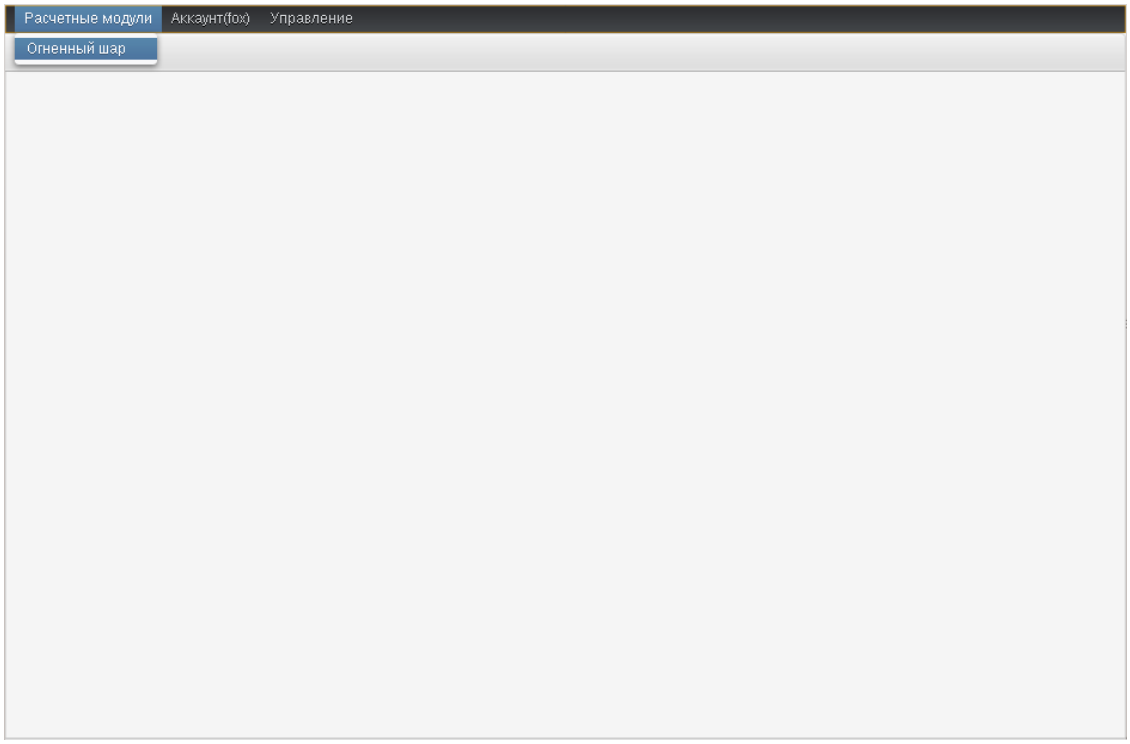


Рис. 1. Рабочая область.

Следующим шагом пользователю необходимо ввести исходные данные (Рис. 2.)

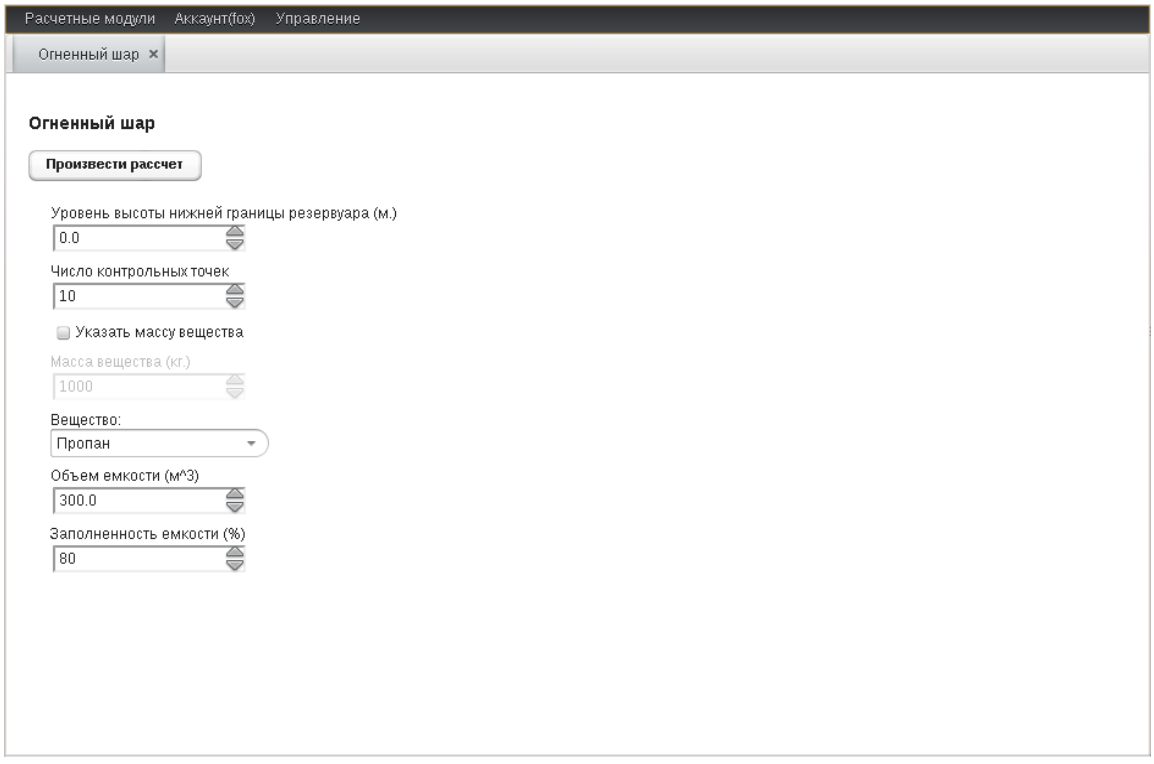
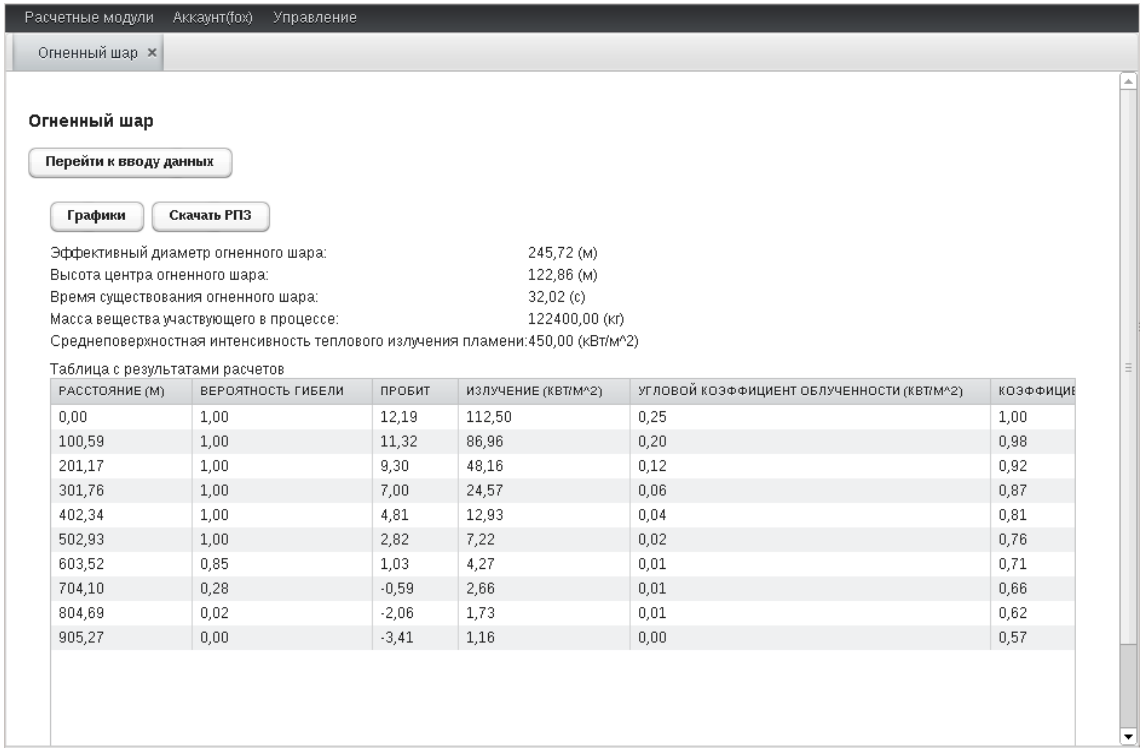


Рис. 2. Ввод исходных данных.

После ввода исходных данных пользователь получает результаты расчетов в табличном виде (Рис.3.).



РАССТОЯНИЕ (М)	ВЕРОЯТНОСТЬ ГИБЕЛИ	ПРОБИТ	ИЗЛУЧЕНИЕ (КВТ/М^2)	УГЛОВОЙ КОЭФФИЦИЕНТ ОБЛУЧЕННОСТИ (КВТ/М^2)	КОЭФФИЦИЕ
0,00	1,00	12,19	112,50	0,25	1,00
100,59	1,00	11,32	86,96	0,20	0,98
201,17	1,00	9,30	48,16	0,12	0,92
301,76	1,00	7,00	24,57	0,06	0,87
402,34	1,00	4,81	12,93	0,04	0,81
502,93	1,00	2,82	7,22	0,02	0,76
603,52	0,85	1,03	4,27	0,01	0,71
704,10	0,28	-0,59	2,66	0,01	0,66
804,69	0,02	-2,06	1,73	0,01	0,62
905,27	0,00	-3,41	1,16	0,00	0,57

Рис.3. Результаты расчетов в табличном виде.

Результаты расчетов могут быть представлены также в виде графиков (Рис.4, Рис.5).

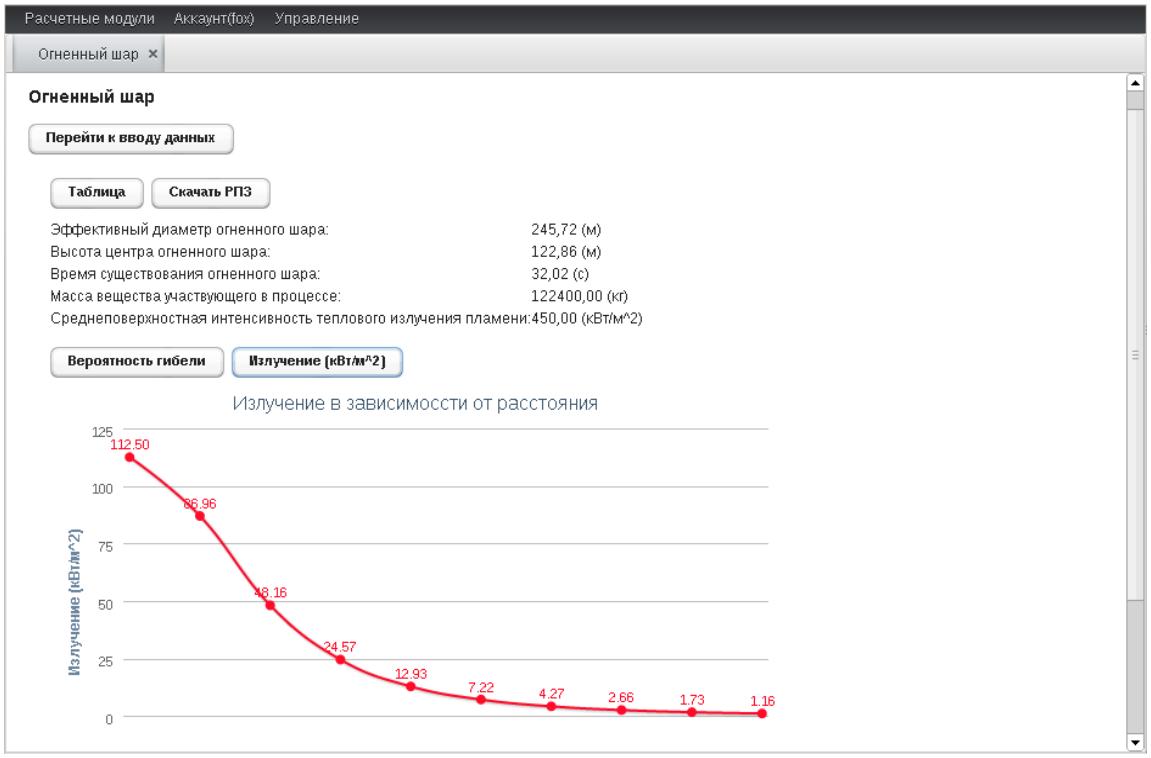


Рис.4. Результаты расчетов в виде графиков.

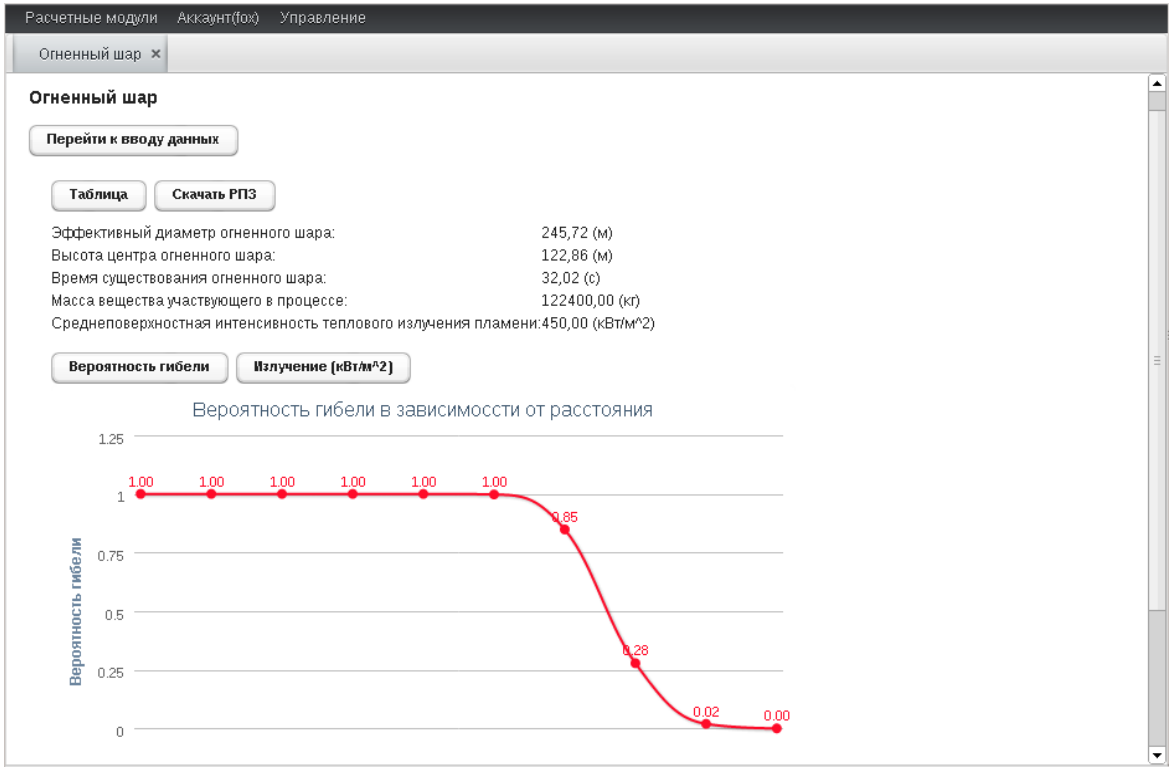


Рис.5. Результаты расчетов в виде графиков.

Список литературы

1. Морозов О.А. Открытая платформа интеграции сервисов "Безопасность в техносфере" // Материалы Международной конференции с элементами научной школы для молодежи «Производство. Технология. Экология» науч. ред.: В. М. Колодкин, И.Л. Бухарина. - Ижевск: Удмурт. ун-т, 2010.
2. http://ru.wikipedia.org/wiki/Бизнес-процесс
3. http://ru.wikipedia.org/wiki/Веб-служба
4. http://www.eclipse.org/
5. http://xstream.codehaus.org
6. http://commons.apache.org/math/
7. http://xstream.codehaus.org/annotations-tutorial.html