Вісник Харківського національного університету
Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»
№ 733, 2006, с.3-16

УДК 681:3.06

Використання jBPM для моделювання бізнес-процесів в розподілених інформаційних системах

М. О. Алєксєєв, Л. С. Глоба, Ю. М. Молчанов
Національний технічний університет України «Київський політехнічний інститут», Інститут телекомунікаційних систем, Україна

Using of jBPM for business-process management of the distributed information telecommunication systems is investigated. The research is based on the experience of organizing the automated information interchange between the software of Central and regional segments of the Chernobyl database of the Ministry of Emergency of Ukraine.

Актуальність використання BPM-технологій

Відомо, що поява нових технологій призводить до дійсно серйозних змін у сфері бізнесу й діловодства. Однією з таких технологій у наш час є автоматизоване керування діловими процесами (BPM).

Одержувати знання й засоби, необхідні для збереження конкурентоспроможності на нових світових ринках, найкраще допомагає співробітництво, формування партнерських відносин. Щоб досягти успіху в цій стратегії, підприємство повинно так представити свій бізнес, щоб залучити потенційних ділових партнерів. При цьому позитивним фактором є відкриття доступу до інформації про свої ділові процеси. Очевидно, що для цього потрібна універсальна мова опису процесів, метод їхнього визначення, так само точний й недвозначний, як мова програмування. З іншого боку, сучасні інформаційні системи, що використовуються підприємствами та компаніями для своїх бізнес-потреб є розподіленими у телекомунікаційному середовищі, складними за своєю архітектурою та інформаційною структурою. Тому дослідження можливостей сучасних BPM-систем щодо моделювання бізнес-процесів в таких системах також є задачею актуальною.

Аналіз існуючих систем керування бізнес-процесами

Оглядовий аналіз сучасних засобів BPM показав [1], що всі вони мають такі функціональні елементи як засоби проектування, движки й засоби моніторингу. По призначенню існуючі рішення BPM діляться на системи підтримки адміністративного керування (Administrative and Task Support), засоби підтримки колективної роботи (Collaborative BPM), прикладні BPM-Компоненти (Application-Specific or Preconfigured BPM), системи, націлені на інтеграцію автоматизованих систем (Integration-Focused BPM). Існують також універсальні рішення (Pure-Play BPM).

Всі існуючі системи BPM ґрунтуються на відповідних стандартах і специфікаціях мов моделювання бізнес-процесів. На даних момент відомо безліч таких стандартів і специфікацій. Найпоширенішими серед них є Business

Process Execution Language (BPEL), Business Process Modeling Language (BPML) [2, 3].

У доступному для огляду майбутньому ні однієї компанії, звичайно, не удасться запропонувати системи, що задовольняють усім вимогам до системи керування бізнес-процесами розподілених інформаційних телекомунікаційних систем. Але вже сьогодні існують ефективні рішення, що забезпечують основні можливості по впровадженню, виконанню, підтримці й оптимізації, які можна використовувати разом з іншими технологіями, приклад з автоматизацією ділових процесів, керуванням ресурсами підприємства й ін., створивши в такий спосіб інформаційну інфраструктуру керування процесами.

Користувачі систем управління процесами повинні мати можливість стежити за ходом процесів на всіх рівнях і будувати консолі керування, щоб удосконалювати процеси. Їм також необхідні інструменти періодичної перевірки наскрізних процесів для пошуку нових можливостей, створення цілком змінених процесів, служб і продуктів. Основна роль тут за емуляцією.

Вимогам інтегрування, відкритості і здатність масштабуватися — задовольняють BPM-системи, реалізовані на платформі J2EE. Платформа J2EE заснована на відкритих, підтримуваних лідерами IT-галузі, стандартах, що сприяє інтеграції з корпоративними програмними засобами. Багаторівнева архітектура J2EE дозволяє досягати високій продуктивності і надійності за рахунок розподілу навантаження між серверами.

Лідером серед таких систем на сьогоднішній день є система JBPM [4], компанії JBoss. JBoss jBPM є гнучкою, розширюваною системою керування технологічним процесом. JBoss jBPM має інтуїтивну мову процесу для вираження ділових процесів графічно в термінах задач, стану чекання для асинхорної передачі, таймерів, автоматизації дій. Для зв'язку цих операцій JBoss jBPM має потужний і розширюваний механізм потоку управління.

Практична апробація системи JBoss JBPM

Для більш детального вивчення можливостей використання jBPM для моделювання бізнес-процесів в росподілених інформаційних системах, була обрана задача організації автоматизованого обміну інформацією між програмним забезпеченням Центрального і регіонального сегментів Чорнобильського банку даних МНС України. У здійсненій реалізації передбачалася можливість обміну документами і базами даних MS Office у виді файлових архівів, з використанням електронної пошти, через протокол http, а також можливість запуску і виконання функцій web-інтерфейсу.

Реалізація моделі розподіленої інформаційно – телекомунікаційної системи

На першому етапі рішення задачі за допомогою графічного проектувальника jBPM, убудованого в пакет розробки Eclipse, був спроектований граф відповідного бізнес-процесу для регіонального сегмента Чорнобильського банку даних МНС України, представлений на рис. 4.1.

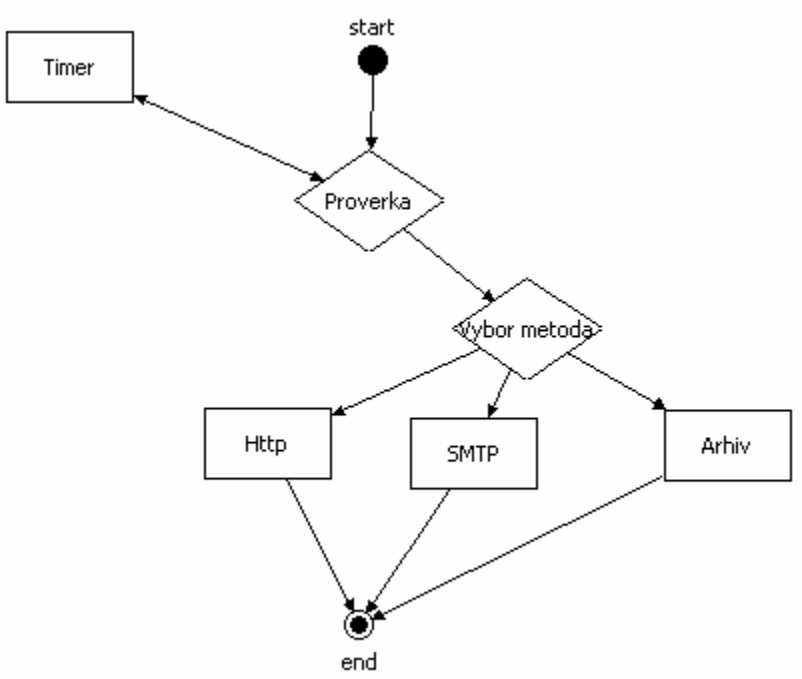

Рис. 4.1 Граф моделі бізнес-процесу регіонального сегмента

Даний граф складається з 8 вузлів і 6 переходів. Граф починається зі стартового вузла. Далі випливає вузол, у якому відбувається перевірка вмісту заздалегідь зазначеної папки на наявність у ній необхідного для продовження роботи файлу. У випадку, якщо такий мається, маркер виконання переходить на наступний вузол. Якщо файл відсутній, то виконання графа переходить у таймер, по витіканню котрого, виконання графа відновляється з моменту перевірки вмісту папки. Далі випливає вузол, на вході якого мається «екшен», що пропонує користувачеві вибрати один із трьох варіантів продовження роботи з файлом:

– заархівувати файл і перемістити в заздалегідь визначений каталог;
– відправити файл за допомогою електронної пошти за допомогою SMTP протоколу;
– передати файл по HTTP протоколу.

Далі відбувається обрана користувачем дія, після чого маркер графа переходить у кінцевий стан.

На другому етапі рішення задачі за допомогою графічного проектувальника jBPM, убудованого в пакет розробки Eclipse, був спроектований граф відповідного бізнес-процесу для центрального сегмента Чорнобильського банку даних МНС України, представлений на рис. 4.2

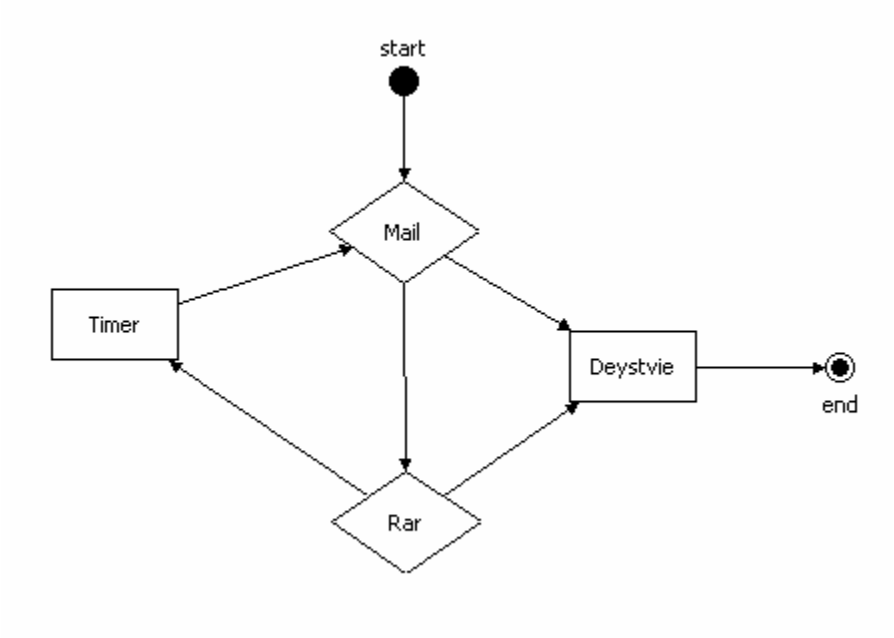

Рис. 4.2 Граф центрального сегмента

Даний граф складається з 6 вузлів і 6 переходів. Граф починається зі стартового вузла. Далі випливає вузол, у якому відбувається перевірка наявності у відповідній поштовій скриньці необхідного для продовження роботи по даніїй гілці графа листа з прикріпленим файлом. У випадку, якщо такий мається, маркер виконання переходить на вузол, у якому виконується необхідне переміщення файлу. Якщо лист відсутній, то виконання графа переходить у вузол, у якому виконується перевірка наявності файлового архіву. Якщо архів відсутній, то маркер переходить у таймер, по витіканню якого виконання графа відновляється з моменту перевірки електронної пошти.

Третій етап практичної частини полягав у додаванні до графів обох сегментів коду, написаного мовою Java з метою надаванню графові необхідних програмних функціональних можливостей

Так, для регіонального сегмента потрібно додати 4 «екшена», реалізованих у виді класів мови Java. Однак на даному етапі виникла складність написання «екшена» реалізуючу функцію вибору користувачем методу доставки файлу в Центральний сегмент, тому що даний «екшен» може бути реалізований тільки мовою BeanShell.

Вихід був знайдений шляхом поділу графа на 3 окремих підграфи, кожний з яких реалізує окремий метод доставки файлу, що відображено на рис. 4.3.

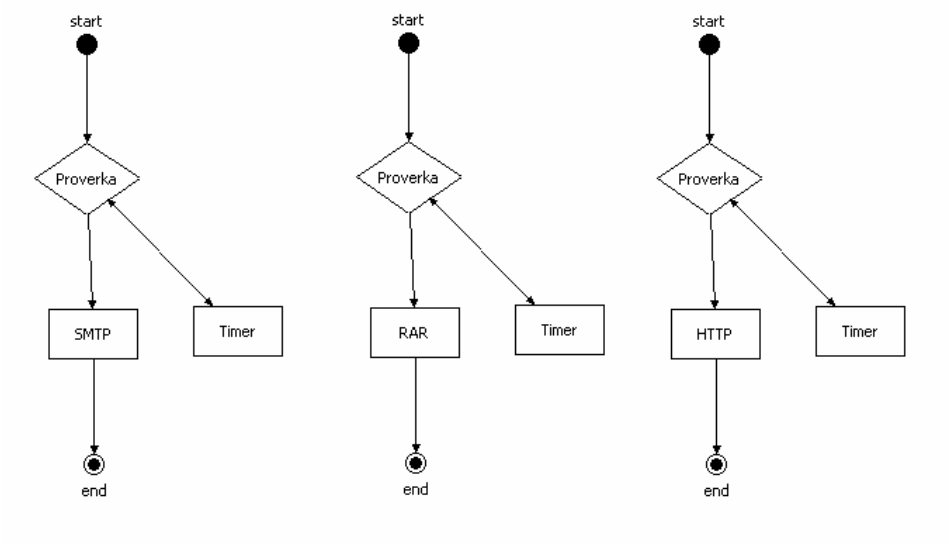

Рис. 4.3 Графи регіонального сегмента

До кожного з них були додані відповідні класи, представлені у відповідних лістингах.

Лістинг коду класа FileHandleException.class

```
public class FileHandleException extends Exception{
    public FileHandleException(String message) {
        super(message);
    }
}
```

Лістинг коду класа FileHandler.class

```
import javax.mail.internet.*;
import javax.mail.Message;
import javax.mail.Transport;
import javax.mail.Session;
import javax.mail.Provider;
import javax.activation.FileDataSource;
import javax.activation.DataHandler;
import java.io.*;
import java.util.Properties;
import java.net.Socket;

public class FileHandler {
```

```
      protected static final String[] commands = new
String[]{"--c --copy /c","--s --send /s","--u --upload
/u","--h --help /h"};

      protected File file;
      protected String dest;
      protected Properties fileProps;

      public FileHandler(File file, String dest) throws
FileHandleException {
          this.file = file;
          this.dest = dest;
          this.fileProps = new Properties();
          try{
              this.fileProps.load(new
FileInputStream("config.properties"));
          } catch (IOException e) {
              throw new FileHandleException("Can not load
configuration file.");
          }
      }

      public static void main(String[] args){
          if(args.length == 3){

              try{

                  File file = new File(args[1]);
                  if(!    file.exists())    throw    new
FileHandleException("File \""+ args[1] +"\" not found.");

                  FileHandler fh = new FileHandler(file,
args[2]);

                  switch(getCommand(args[0])){
                      case 0: fh.copyFile(); break;
                      case 1: fh.sendFile(); break;
                      case 2: fh.uploadFile(); break;
                      default: showHelp();
                  }

              }catch(Exception e){
                  pl(e.getMessage());
                  pl("\nUse \"java FileHandler -h\" for
help.");
              }
          }
```

```
            else{
                showHelp();
            }

        }

        public static int getCommand(String cmd){
            for(int i = 0; i < commands.length; i++)
                if(commands[i].indexOf(cmd.toLowerCase())
>= 0) return i;

            return -1;
        }

        public static void p(Object o){      if(o !=null)
System.out.print(o.toString());   }
        public static void pl(Object o){     if(o !=null)
System.out.println(o.toString());   }

        public static void showHelp(){
            String msg = "\nCopies one file to another
location, sends by e-mail or uploads to file server via
TCP/IP connection.\n" +
                    "\nUsage: java FileHandler [command]
[file] [destination]\n" +
                    "\n  command:\t" +
                    "\n   \t-c copy  \t" + "Copies file to
destination on local system." +
                    "\n  \t-s send  \t" + "Sends file by e-
mail      (need      to      specify      SMTP-host      in
\"config.properties\")."+
                    "\n   \t-u upload\t" + "Uploads file to
the server."+
                    "\n    \t-h help    \t" + "Shows this
message."+
                    "\n   file          \t" + "Specifies the
file to be hanled." +
                    "\n    destination\t" + "Specifies the
directory for the new file, e-mail for sending or host
for uploading." +
                    "\n\nExample:   java   FileHandler   -u
\"C:\\My Documents\\Read Me.txt\" 127.0.0.1";
            pl(msg);
        }


        public void copyFile() throws FileHandleException {
```

```
          try{
              InputStream         in         =         new
BufferedInputStream(new FileInputStream(file));
              OutputStream        out        =         new
BufferedOutputStream(new FileOutputStream( new File(dest,
file.getName()) ));
              inToOut(in, out);
              pl("File was successfully copied.");
          }catch(Exception e){
              throw new FileHandleException("Can not copy
file. Reason: " + e.getMessage());
          }
      }



      public void sendFile() throws FileHandleException {

          try{

              String                host                =
fileProps.getProperty("mail.smtp_host");
              String              mailfrom              =
fileProps.getProperty("mail.from");
              String              mailtext              =
fileProps.getProperty("mail.text");
              if(mailtext == null)
                  mailtext = "";
              String               subject              =
fileProps.getProperty("mail.subj");
              if(subject == null)
                  subject = "File sedned by application
\"FileHandler\"";

              Properties props = new Properties();

props.put("mail.transport.protocol","smtp");
              props.put("mail.smtp.host", host);

              Session  sess  =  Session.getInstance(props,
null);

              MimeMessage msg = new MimeMessage(sess);

              msg.setFrom(new    InternetAddress(mailfrom)
);
```

```
                msg.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(dest, false));
                msg.setSubject(subject);

                MimeBodyPart textPart = new MimeBodyPart();
                textPart.setContent(mailtext,   "text/plain;
charset=UTF-8");

                FileDataSource       fds       =       new
FileDataSource(file);
                MimeBodyPart attach = new MimeBodyPart();
                attach.setDataHandler(new
DataHandler(fds));
                attach.setFileName(fds.getName());

                MimeMultipart      content      =      new
MimeMultipart();
                content.addBodyPart( textPart );
                content.addBodyPart( attach );
                msg.setContent(content);

                Transport.send(msg);

                pl("File was sended by e-mail.");


            } catch (AddressException e) {
                throw new FileHandleException("Wrong e-mail
address.");
            } catch(Exception e){
                throw new FileHandleException("Can not send
e-mail, check your mail configuration.");
            }

        }

        public void uploadFile() throws FileHandleException
{

            try{
                Socket    socket    =    new    Socket(dest,
Integer.parseInt(fileProps.getProperty("fileserver.port")
));

                OutputStream              out              =
socket.getOutputStream();
```

```
              InputStream         in         =         new
BufferedInputStream(new FileInputStream(file));
              OutputStreamWriter    nameWriter    =    new
OutputStreamWriter(out);
              nameWriter.write(file.getName()+"\n");
              nameWriter.flush();

              inToOut(in, out);

              pl("File was successfully uploaded.");
           }catch(Exception e){
              throw                                        new
FileHandleException(e.getMessage());
           }
      }


      protected void inToOut(InputStream in, OutputStream
out) throws IOException {
            byte[] buffer = new byte[1024];
                   int len = 0;
                   try
                   {
                       while ((len = in.read(buffer)) > 0)
                       {
                           out.write(buffer, 0, len);
                       }
                   }
                   finally
                   {
                       out.close();
                       in.close();
                   }
      }

  }
```

Лістинг коду класа SimpleFileServer.class

```
  import java.net.ServerSocket;
  import java.net.Socket;
  import java.io.*;

  public class SimpleFileServer {

      public static void main(String[] args){
          ServerSocket ss;
```

```
          Socket socket;
          try{
              Integer port = Integer.parseInt(args[0]);
              File incoming = new File(args[1]);

              try{
                  if(!incoming.exists()                 ||
!incoming.isDirectory())
                      throw      new      Exception("wrong
uploading path.");

                  pl("File server started...");
                  while(true){
                      ss = new ServerSocket(port);
                      socket = ss.accept();

                      pl("Client       connected:       "+
socket.getRemoteSocketAddress());

receiveFile(socket.getInputStream(), incoming);

                      socket.close();
                      ss.close();
                  }
              }catch(Exception e){
                  pl("\nError:  "+e.getMessage());
              }
          }catch(Exception e){
              pl("\nUse:    java  SimpleFileServer  [port]
[uploading path]\n" +
                      "Example:    java    SimpleFileServer
2909 C:\\MyServer\\incoming");
          }

      }

      public static void p(Object o){      if(o !=null)
System.out.print(o.toString());   }
      public static void pl(Object o){     if(o !=null)
System.out.println(o.toString());   }

      protected static void receiveFile(InputStream in,
File incoming) throws Exception {

          BufferedReader bin
                  =       new       BufferedReader(new
InputStreamReader(in));
```

```
            String filename = bin.readLine();
            OutputStream out = new BufferedOutputStream(new
FileOutputStream( new File(incoming, filename )));

             byte[] buffer = new byte[1024];
                    int len = 0;
                    try
                    {
                        while ((len = in.read(buffer)) > 0)
                        {
                            out.write(buffer, 0, len);
                        }
                    }
                    finally
                    {
                        out.close();
                        in.close();
                    }

            pl("Received file: " + filename);
        }
    }
```

У результаті програвання моделі регіонального сегмента були отримані результати, представлені на малюнках 4.4 , 4.5 , 4.6

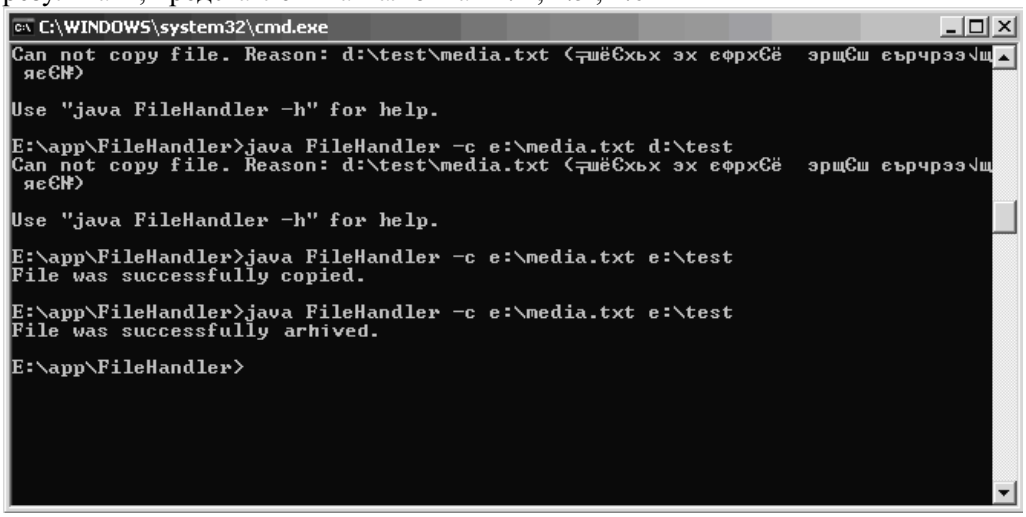

Рис. 4.4 Результат програвання гілки графа у випадку вибору користувачем методу архівування

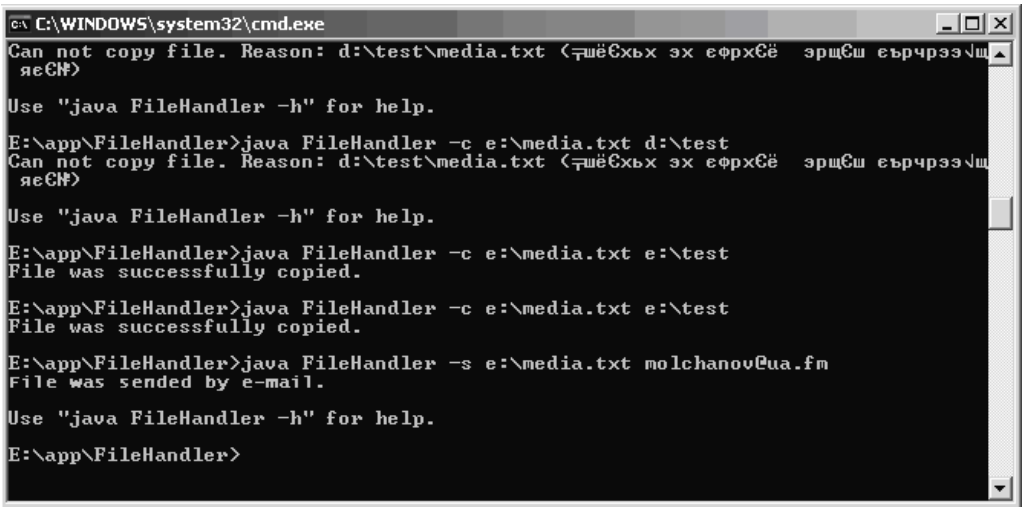

Рис. 4.5 Результат програвання гілки графа у випадку вибору користувачем методу пересилання по електронній пошті з використанням протоколу SMTP

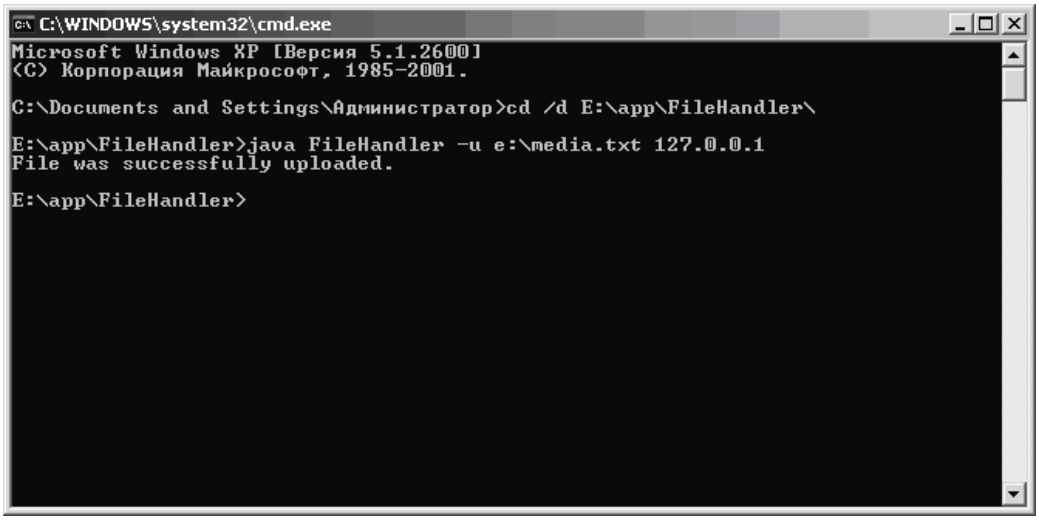

Рис. 4.6 Результат програвання гілки графа у випадку вибору користувачем методу пересилання с допомогою протоколу HTTP

На сервері центрального сегмента був запущений HTTP файл-сервер, написаний мовою Java. Даний сервер працює у фоновому режимі і призначений для доставки відправленого з регіонального сегмента файлу і переміщення цього файлу в заздалегідь визначену в конфігураційному файлі директорію

У результаті програвання моделі центрального сегмента були отримані результати, представлені на рис. 4.7 .

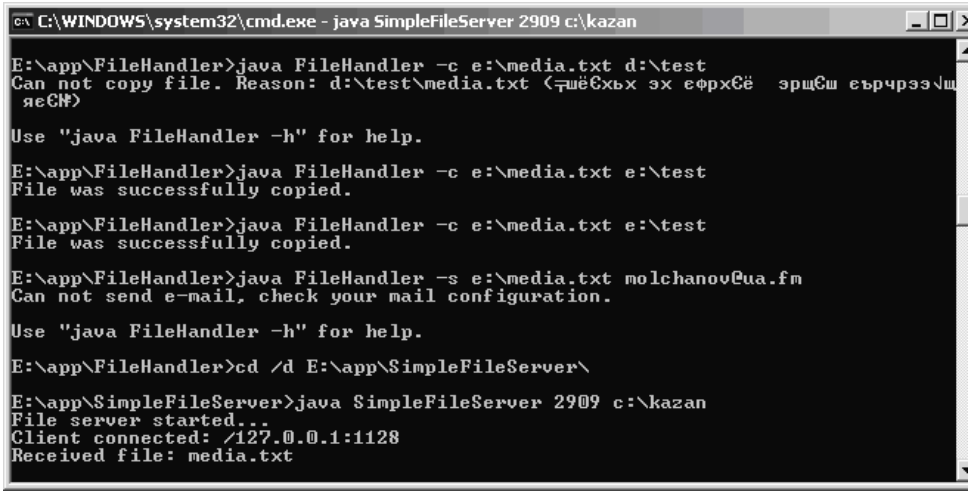

Рис. 4.7 Результат роботи HTTP сервера центрального сегмента

Висновки по даному дослідженню й перспективи подальших досліджень

Досвід, отриманий у процесі використання JBPM показав, що, зокрема, завдяки використовуваним Java і XML технологіям, система є такою, що розвивається і перспективною. JBPM є машино-незалежною і об`єктно-орієнтованою, порівняно з розглянутими у оглядовій частині дипломної роботи, легкою у використанні професіональними програмістами. Однак можна зробити висновок, що на даному етапі свого розвитку система є складною для використання менеджерами, що не знайомі з технологією Java у досконалості, оскільки надає дружній інтерфейс тільки в убудованому графічному проектувальнику, що сам по собі дозволяє лише створювати графічне представлення послідовності бізнес процесу. Більш деталізоване моделювання інформаційних систем на рівні форм і функцій (об'єктів і методів) можливо тільки при участі Java-програмістів. Також слід зазначити відсутність у jBPM прийнятних засобів для моделювання бізнес-процесів розподілених інформаційних систем, що робить складним розробку корпоративних інформаційних систем рівня підприємства, призначених для роботи в розподіленому інформаційно-телекомунікаційному середовищі.

В розширеній версії моделі регіонального сегменту рекомендується використовувати паралельний граф з функцією вибору наступної дії користувачем, написаній на скриптовій мові BeanShell. Оскільки робота виконана в операційній системі Windows 2003 Server, рекомендується провести фінальний тест моделі на Unix-подібних операційних системах, а також на MacOS.

ЛІТЕРАТУРА

1. Что такое BPMS. - http://www.bpms.ru/intro/
2. Спецификация языка Business Process Execution Language – http://bpel.com
3. Спецификация языка Business Process Modeling Language – http://bpmi.org
4. Ю.Н.Молчанов, Авторский перевод документации Jboss jBPM – http://citforum.ru