Implementation of Distributed M-Learning Applications Using WF and WCF

Paul POCATILU
Bucharest Academy of Economic Studies
ppaul@ase.ro

The development of distributed e-learning and m-learning applications involves several tools and technologies. In order to provide a common way to access the learning services in a distributed environment for both desktop and mobile clients, standardized solutions have to be used. One direction is represented by Web applications. Another direction is given by the Web services. In this paper is presented a solution that involves the use of Windows Workflow Foundation (WF) and Windows Communication Foundation (WCF) frameworks to develop distributed e-learning applications.
Keywords: *E-learning, M-learning, Workflows, WCF, WF*

1 Introduction

Mobile learning applications involve the use of mobile devices in learning processes. The main characteristics of mobile applications are discussed in [1]. The evolution of knowledge-based society involves the development of educational process through a distributed mobile learning environment.

The development of mobile learning distributed applications has specific characteristics. Distributed platforms have a similar architecture as Web-based platforms, but the client application is a rich application and not a simple mobile Web browser. The advantages of this platform are [9]:

- Rich user interface;
- Support for multimedia content;
- E-learning content can be easily updated on the server;

There are also some disadvantages:

- The user need to install and setup the client application;
- The user have to learn how to use the application;
- Possible additional costs for traffic usage.

In [7] is presented the process of distributed mobile learning applications development. The e-learning systems based on cloud computing are analyzed in [2].

A single development approach for both mobile and desktop clients has many benefits for the e-learning systems. This paper focuses on a specific implementation of e-learning distributed applications using Workflow Foundation and Windows Communication Foundation technologies.

The paper is structured as follows:

Section *WF Overview* presents the Windows Workflow Foundation and analyzes the activities available on Windows Workflow Foundation used to send and receive data over a computer network.

In section *Distributed E-learning Applications Architecture* is proposed an architecture for a distributed application based on WF and WCF.

Workflow Services presents two workflows: one based on Web services and the other based on WCF.

The last section, *Server and Client Implementation*, provides an example of a service.

The paper ends with conclusions and future work.

2 WF overview

Windows Workflow Foundation is a technology that provides support for application development based on workflows and rules. The activities represent the components of a workflow. Workflows can be defined using an integrated designer or using XOML files. A workflow can be sequential, state based, flowchart or custom. Figure 1 depicts available activities for .NET 3.5.

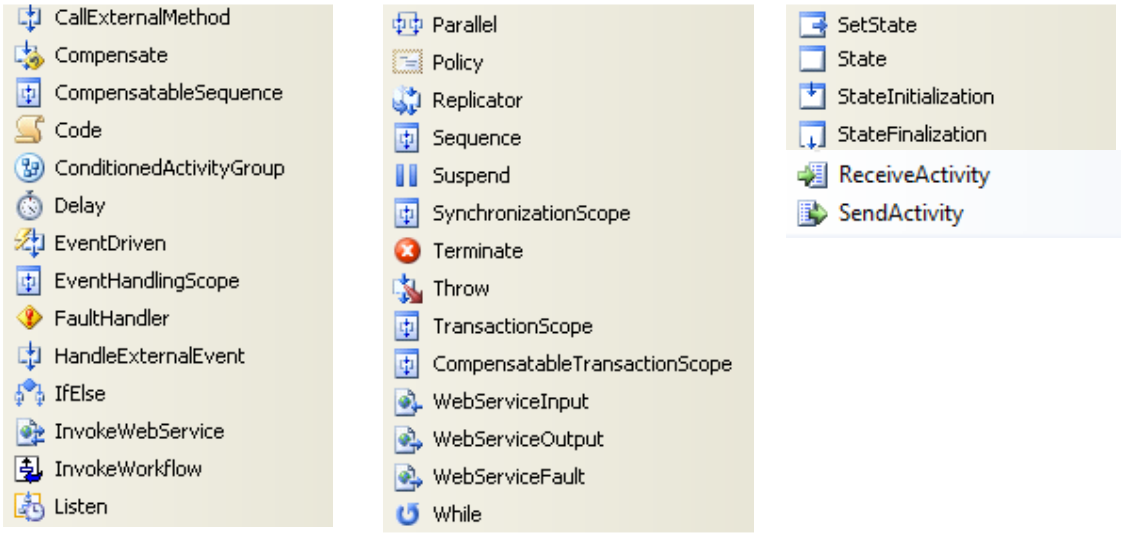

Fig. 1. WF available activities

Custom activities can be implemented in libraries and added to the projects and available from the Toolbox. Conditions can be used in *IfElseBranch*, *While*, *Replicator* and *ConditionedActivityGroup* activities.

A condition can be coded or specified as a rule. A host process (a desktop application or ASP.NET worker process) must load and launch the workflow runtime before starting a workflow.

Microsoft .NET Framework 4.0 introduces new activities and the possibility to run a workflow without a runtime. The WF 4.0 activities are presented in Table 1.

Table 1. WF activities for .NET 4.0

Category	**Activity**	**Category**	**Activity**
Control Flow	DoWhile	**Runtime**	Persist
	ForEach		TerminteWorkflow
	If	**Primitives**	Assign
	Parallel		Delay
	ParallelForEach		InvokeMethod
	Pick		WriteLine
	PickBranch	**Transaction**	CancellationScope
	Sequence		CompensableActivity
	Switch		Compensate
	While		Confirm
Flowchart	Flowchart		TransactionScope
	FlowDecision	**Collection**	AddToCollection
	FlowSwitch		ClearCollection
Messaging	CorrelationScope		ExistsInCollection
	InitializeCorrelation		RemoveFromCollection
	Receive	**Error Handling**	Rethrow
	ReceiveAndSendReply		Throw
	Send		TryCatch
	SendAndReceiveReply	**Migration**	Interop
	TransactedReceiveScope		

Workflows can be published as Web services, WCF services or available as libraries (dll

files) or included in applications. External calls can be made from an activity, either from dynamic libraries either from Web services.

The workflow engine can be used for Web based applications and desktop applications. Workflow designer can be hosted in desktop applications only.

The workflows can be defined:

- with Workflow designer;
- using XAML (Extensible Application Language) in XOML files (declarative workflows);
- dynamically in code.

Conditions can be hard-coded or use Rules editor or engine.

Windows Workflow Foundation includes several activities used for communication. Depending on the NET Framework version, there are various classes that provide support for communication in Windows Workflow Foundation.

Web services methods can be called using *WebServiceInputActivity* and *WebServiceOutputActivity* classes. The *WebServiceInput* activity role is to receive the input from a specific method of a Web Service. *WebServiceOutput* activity is used to send data to a Web service as a response of a method call. *WebServiceOutput* activity cannot be used without a *WebServiceInput* activity.

Starting with .NET Framework 3.5 support for Windows Communication Foundation is included through *SendActivity* and *ReceiveActivity* [4]. In order to access a workflow as a service, *ReceiveActivity* is used. It is based on a contract that defines all operations published by the service. *SendActivity* is used to consume a service within the workflow.

.NET Framework 4.0 adds new communication related activities to Windows Workflow Foundation [3].

Send activity is used to send messages and *Receive* activity to receive messages. *SendAndReceiveReply* includes *Send* and *ReceiveReply* activities. They are used when a message is sent within the workflow and a response is required. *ReceiveAndSendReply* includes *Receive* and *SendReply* activities and is used when a received message need a response.

Message correlation is possible with *CorrelationScope* and *InitializeCorrelation* activities.

3 Distributed E-learning Applications Architecture

According to [10], the architecture of distributed and collaborative learning system includes the followings components: course authoring system, learning management system and learning content management system. These components are presented in Figure 2, adapted for mobile learning systems.

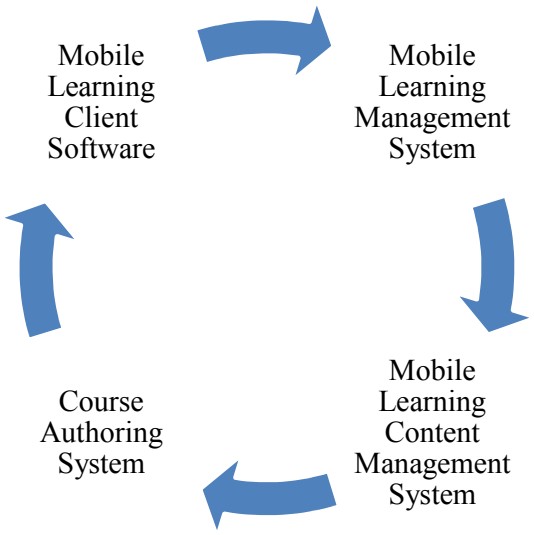

Fig. 2. The functional model of a distributed mobile learning system

The functional model of collaborative learning system provides a visual representation of components integrated and the flows between them. The integration of these components of a collaborative mobile learning system is depicted in Figure 3.

Mobile Learning Content Management System (MLCMS) includes the functionality for create, read, update and delete the mobile learning content.

Course Authoring System (CAS) helps in content development and it should be adapted for mobile content having in mind the limitations of such devices. The trainer or a specialist uses the CAS to create mobile content for the learning system. If the learning content does not require a specific format

(like multimedia, animation etc.) the MLCMS can be used.
Mobile Learning Management System (MLMS) has the role to manage the user interaction with the mobile learning system: courses, tests, quizzes and other activities.

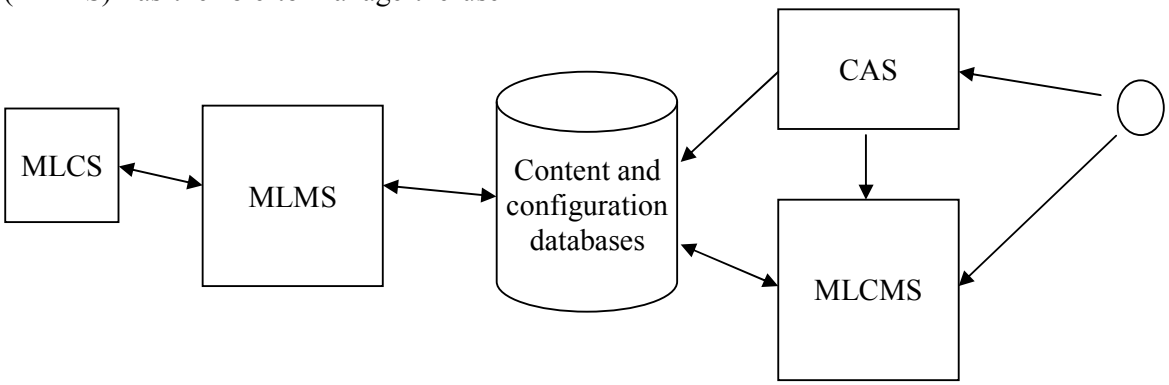

Fig. 3. The integration of collaborative mobile learning system components

The Mobile Learning Client Software (MLCS) runs on a mobile device and is usually adapted for a specific operating system (Android, Bada, iOS, Symbian, Windows Mobile, Windows Phone etc.), and/or platform (Java ME, .NET CF, Qt etc.). MLCS connects to a MLMS and, depending on the implementation, includes or not logic. If the content on the mobile device is loaded from a mobile web browser, the entire logic exists on the MLMS.
The presented application is based on WF and WCF and other Microsoft technologies. The e-learning application logic will be implemented using Windows Workflow Foundation. In order to be available to client applications over the network, the services are published as Web Services or WCF services. Clients are either mobile or desktop applications. They connect to Web services or WCF services and they can be implemented using other technologies that Microsoft. In [5] is presented a solution to connect an Android based application to a Web service (developed using ASP.NET) in an e-learning system.

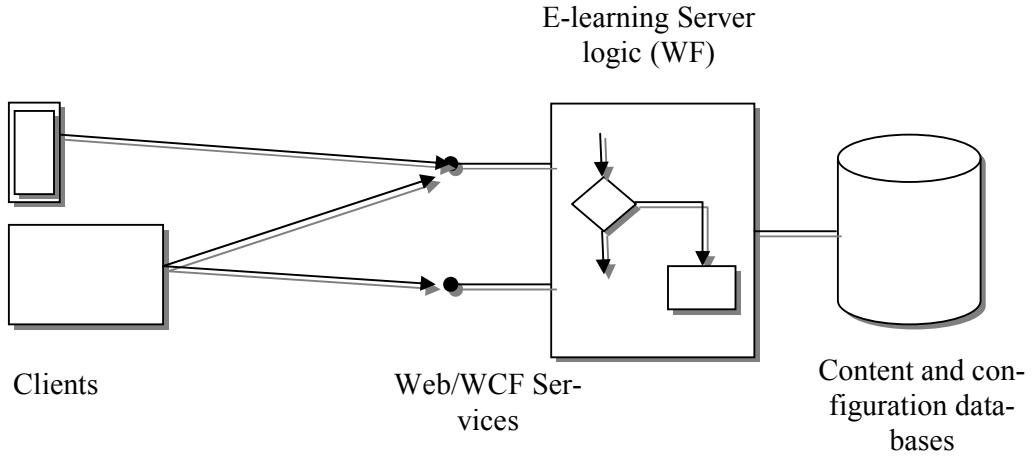

Fig. 4. Distributed e-learning applications architecture using WF and WCF

The databases include all required configuration data, the mobile learning and e-learning content, user data and other specific data used for WF persistence and tracking. Figure 4 depicts the e-learning application architecture.

4 Workflow Services

The entire application logic is implemented using Windows Workflow Foundation.

As an example is presented the modules responsible for file uploading and downloading. There are two implementations: one that exposes methods of a Web services and the other the methods are through a WCF service.

In order to assure the workflow persistence on the server side, the *SqlWorkflowPersistenceService* is used. In order to do that, a database needs to be configured (SQL Server or SQL Express). The path %WINDIR%\Microsoft.NET\Framework\v3.0\Windows Workflow Foundation\SQL\EN contains all the necessary scripts to set up the persistence database (*SqlPersistenceService_Schema.sql* and *SqlPersistenceService_Logic.sql*).

The following tag is added on the web.config file under the Services branch:

```
<add
type="System.Workflow.Runtime.Hosting.SqlWorkflowPersistenceService,
System.Workflow.Runtime, Version=3.0.00000.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" UnloadOnIdle="true"/>
```

and to locate the database a connection string section is used:

```
<CommonParameters>
<add name="ConnectionString" value="Initial Catalog=WorkflowPersistenceStore; Data Source=localhost; Integrated Security=true"/>
</CommonParameters>
```

A dedicated user can be set to access this service (its authentication settings can be passed on the connection string, or the NT AUTHORITY\NETWORK SERVICE user can be used. Anyhow, each user has to have the *state_persistence_roles* role (defined on the *WorkflowPersistenceStore* database by the previous scripts).

Figure 5 depicts the design of a workflow used to upload and download files. Behind every *WebServiceInput* and *WebServiceOutput* activity is a function, defined in an interface, that implements specific logic.

All Web services related activities are implemented using an interface defined as a contract to outside world. The service is published and it can be consumed using standardized technologies.

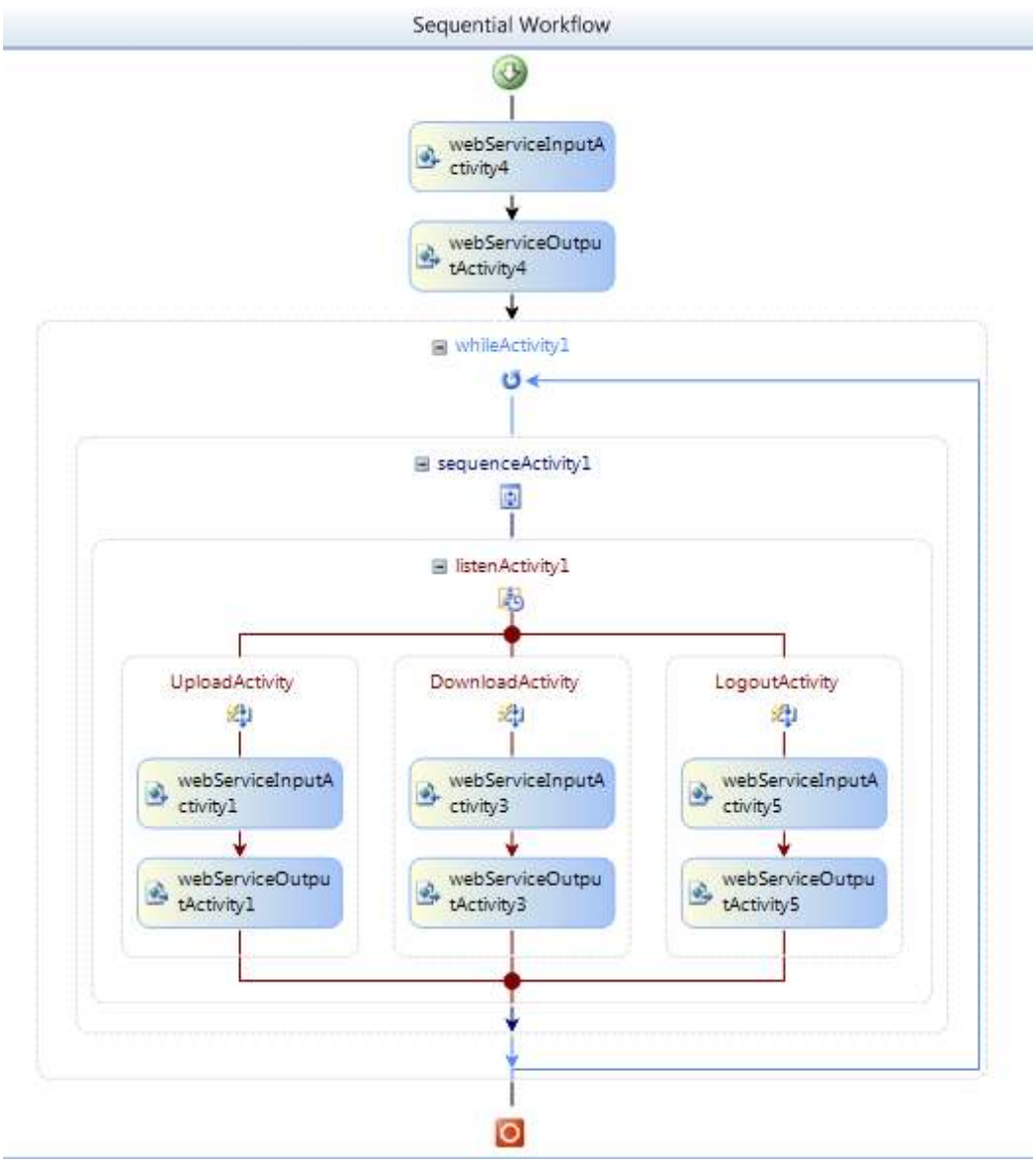

Fig. 5. Sample workflow design based on Web Services

Figure 6 depicts the workflow of the service implemented only for WCF clients. The upload and download activities are based on Receive activity.

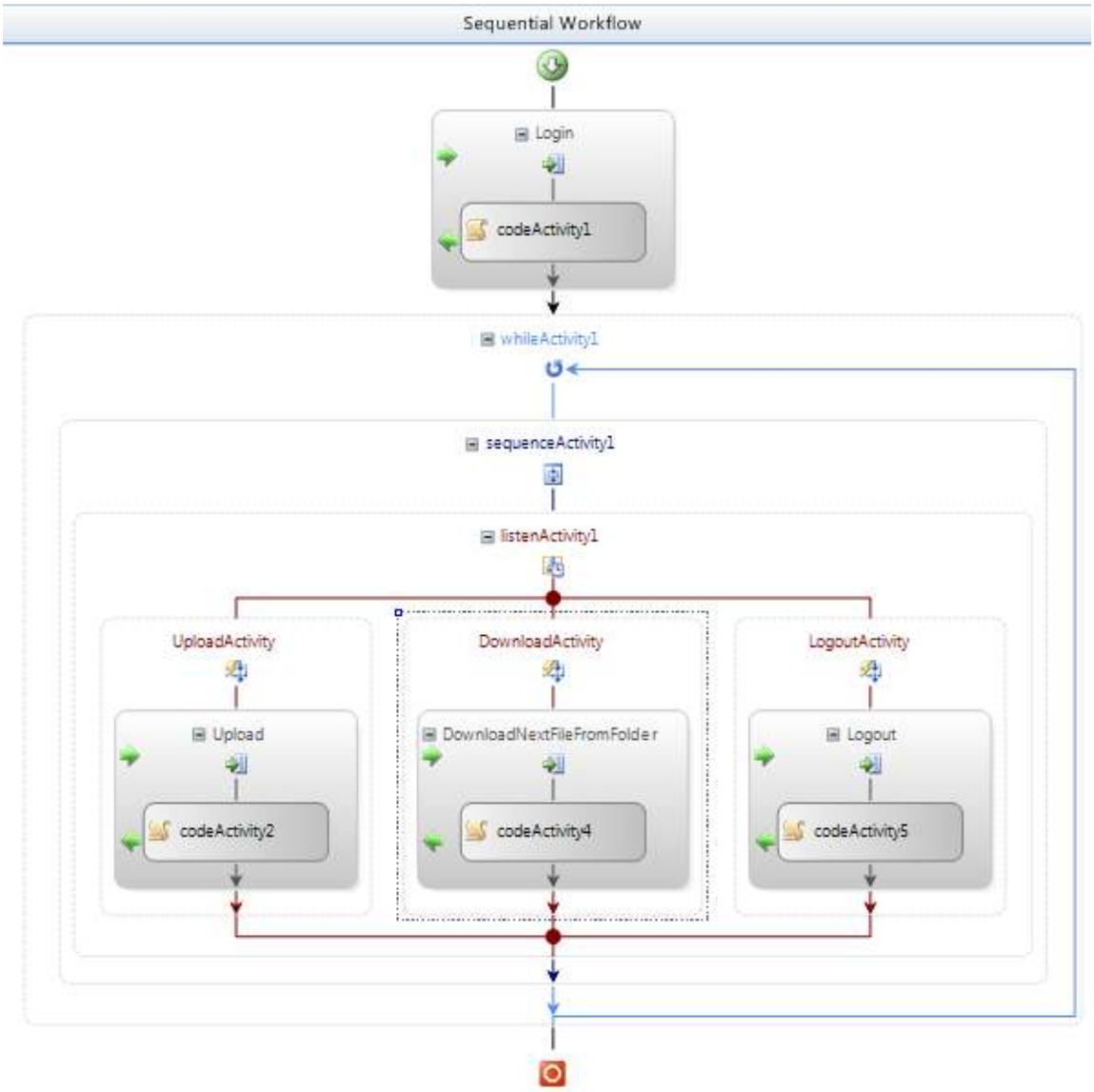

Fig. 6. Sample workflow design based on WCF

The workflows are similar, the difference is given by the way the clients connect and communicate with the services.

5 Server and Client Implementation

On the server side, the service implements methods based on contract provided by an interface. The interface for download and upload service exposes several functions:

```
[ServiceContract]
public interface MLServer
{
  [OperationContract]
  int  Upload(byte  []  file,  string
name);
  [OperationContract]
  byte [] DownloadFile(string name);
  [OperationContract]
  byte[]     DownloadNextFileFromFold-
er();

}
```

The following listing is associated to *Download* function. The function retrieves all the files from a given directory. In the workflow from figures 5 and 6, the function is associated to *DownloadNextFileFromFolder* method.

```
private void Download(object sender, EventArgs e)
{

  if (dirInfo == null)
  {
   dirInfo = new DirectoryInfo(SOURCE_PATH);
   allFiles = dirInfo.GetFiles();
   fileCount = -1;
  }

  areMoreFiles = (++fileCount < allFiles.Length);

  if (areMoreFiles == false)
  {
      webServiceOutputActivity3__ReturnValue_1 = null;
  }
  else
      using (FileStream fs = new FileStream(allFiles[fileCount].FullName,
       FileMode.Open, FileAccess.Read, FileShare.Read))
  {

      Byte[] img = new Byte[fs.Length];
      fs.Read(img, 0, Convert.ToInt32(fs.Length));
      webServiceOutputActivity3__ReturnValue_1 = img;
      fs.Close();
   }
}
```

The implementation of *Upload* function is similar:

```
private void Upload(object sender, EventArgs e)
{
      try
      {
       using (FileStream fs = new FileStream(UPLOAD_PATH +
         webServiceInputActivity1_name1, FileMode.Create))
       {
            fs.Write(webServiceInputActivity1_img2, 0,
               webServiceInputActivity1_img2.Length);
            fs.Close();
       }

       webServiceOutputActivity1__ReturnValue_1 = 1;
      }
      catch (Exception ex)
      {
         webServiceOutputActivity1__ReturnValue_1 = 0;
         throw ex;
      }
}
```

The clients consume the services in a transparent way for the user. In order to download files, the *DownloadNextFileFromFolder* method is called repeatedly. This in an example of such a function call:

```
void Download()
{
      byte[] img = null;
```

```
      img = service.DownloadNextFileFromFolder();

      if (img != null)
      {

      using (FileStream fs = new FileStream(DOWNLOAD_PATH +
      currentProcess.Id.ToString() + DateTime.Now.Ticks.ToString() +
      Thread.CurrentThread.Name, FileMode.Create))
      {
        fs.Write(img, 0, img.Length);
        fs.Close();
      }

      lock (count)
      {
         count.fileCountDownload++;
         count.fileSizeCountDownload += img.Length;
      }
  }
}
```

The *Upload* function is called from the client side very easily as can be seen from the following code:

```
void CallUpload()
{

  byte[] img = null;
  DirectoryInfo dirInfo = new DirectoryInfo(SOURCE_PATH);
  FileInfo[] allFiles = dirInfo.GetFiles();

  foreach (FileInfo file in allFiles)
  {
     using (FileStream fs = new FileStream(file.FullName, FileMode.Open,
      FileAccess.Read, FileShare.Read))
    {

     lock (count)
     {
       count.fileCountUpload++;
       count.fileSizeCountUpload += file.Length;
     }

     img = new Byte[fs.Length];
     fs.Read(img, 0, Convert.ToInt32(fs.Length));
     fs.Close();
      service.Upload(img, currentProcess.Id.ToString() +
        DateTime.Now.Ticks.ToString() + Thread.CurrentThread.Name +
        file.Name);
    }
  }
}
```

In this call example, the files are read from the disk and an array of bytes is initialized with file content. The array is passed to the *Upload* function available on the service. The performances of both methods were measured (WF and Web services and WF and WCF) and there were better for the services consumed using WCF.

6 Conclusions and future work

The development of mobile applications is not a trivial task. There are various platforms

and technologies to choose from. Mobile devices resources are limited. Trying to standardize the way the distributed mobile learning applications communicate with servers will lead to a better integration and better applications.

Using Web services for mobile learning applications helps the process of development by providing a standardized way of communication between mobile clients and servers.

Windows Workflow Foundation and Windows Communication Foundation represent an option for the development of e-learning and m-learning applications.

By using WF and WCF the developers will focus on the e-learning system logic than the implementation, especially with the latest changes in WF.

Acknowledgements

This work was supported by CNCSIS – UEFISCSU, project number PNII – IDEI 2637/2008, project title: *Project management methodologies for the development of mobile applications in the educational system*.

References

[1] D. S. Metcalf II and J. M. De Marco, *mLearning: Mobile Learning and Performance in the Palm of Your Hand*, HRD Press, Inc., 2006.

[2] P. Pocatilu, F. Alecu and M. Vetrici, "Measuring the Efficiency of Cloud Computing for E-learning Systems," *WSEAS Transactions on Computers,* Issue 1, Vol. 9, January 2010, pp. 42-51.

[3] M. J. Collins, *Beginning WF: Windows Workflow in .NET 4.0*, Apress, New York, 2010

[4] B. Bukovics, *Pro WF: Windows Workflow in .NET 3.5*, Apress, 2008

[5] P. Pocatilu, "Developing Mobile Learning Applications for Android using Web Services," *Informatica Economică*, Vol. 14, No. 3, 2010, pp. 106-115

[6] The Developer's Guide | Android Developers [Online]. Available at: http://developer.android.com/guide/index.html (March 2010)

[7] A. Visoiu, "Project Management Methodology for the Development of M-Learning Web Based Applications," *Informatica Economica*, vol. 14, no. 3, 2010, pp. 75-85

[8] C. Ciurea, "The Development of a Mobile Application in a Collaborative Banking System," *Informatica Economică*, Vol. 14, No. 3, 2010, pp. 86-97

[9] P. Pocatilu, M. Doinea, C. Ciurea, "Development of Distributed Mobile Learning Systems," *Proc. of 9th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing (CSECS '10)*, Atena, Grecia, 2010, pp. 196-201

[10] K. Bahrami, M. Abedi and B. Daemi, "Implementation of Distributed E-Learning System on Power Line Network," *Proc. of The Third Advanced International Conference on Telecommunications*, AICT 2007, 13-19 May, 2007, pp. 29.

Paul POCATILU graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 1998. He achieved the PhD in Economics in 2003 with thesis on Software Testing Cost Assessment Models. He has published as author and co-author over 45 articles in journals and over 40 articles on national and international conferences. He is author and co-author of 10 books, (Software Testing Costs, and Object Oriented Software Testing are two of them). He is associate professor in the Department of Economic Informatics of the Academy of Economic Studies, Bucharest. He teaches courses, seminars and laboratories on Mobile Devices Programming, Economic Informatics, Computer Programming and Project Management to graduate and postgraduate students. His current research areas are software testing, software quality, project management, and mobile application development