Framework for Migration of Legacy Applications to Cloud Computing

M2011MM078 Manjula Upashantha

Supervisor Mikio Aoyama

1. Introduction

The author proposes a framework for modernization of legacy applications. In migration, legacy application is moved into a new state and a new platform without changing its core business logical processing. Hence with the migration, cost of development and bugs occurrences should be minimized. The proposed method facilitates migration of legacy application to cloud computing via Service-Oriented Architecture (SOA).

1.1. Background of the Research

Service-orientation is a design paradigm to build computer software in the form of services. Like other design paradigms, e.g. object-orientation, service-orientation provides a governing approach to automate business logic as distributed systems. Similarly cloud computing is the result of an evolution of the widespread adoption of virtualization, SOA, autonomic, and utility computing. Details such as the location of infrastructure or component devices are unknowns to most end-users, who no longer need to thoroughly understand or control the technology infrastructure that supports their computing activities.

Modifing legacy applications, refered as migration to new systems, has few methods. In any methods, there are some challenges to overcome, Generally legacy applications have tighly coupled their business logics with user interface. In cloud computing envirenment exsisting user interfaces are not useful and new interfaces have be built which can run on Web browsers. To address the issues with user interfaces, there are some methods porposed over the past years. Later, some of those proposals are explained in detail.

1.2. Research Problem

One of the key characteristics of legacy applications is its functions are tighly coupled with GUI. Coupling of functions with GUI gives more benefits for the application developers. But when migrating these applications into cloud computing, this coupling makes the job more complicated. Web pages take place of GUI of legacy applications in cloud computing environment. Therefore, decoupling GUI from functions and making functions more independent are challenging task.

2. Related Works

To facilitate migration of the legacy applications into cloud computing, there are many researches going on the last decade. Among those researches, there are interesting methods to address the issues with user interface when migrating the legacy applications into cloud computing.

Sneed et al. have presented a tool supporting process to cut out selected sections of legacy code, and providing them with an XML interface [2]. Stroulia, et al. have proposed a method named CelLEST [3]. A wrapping method is proposed by Canfora, et al. from RCOST (Research Centre on Software Technology) [1]. A generic black-box approach is proposed by Schulze et al. to access legacy applications as cloud-based service [4].

3. Approach

This paper proposes a framework which consists of a set of methods to migrate legacy applications into cloud computing. And this migration is done via SOA. Any of these methods never change or modify legacy functions. Instead they add additional function call gateway to access the legacy functions. This gateway function acts as an interface to the legacy function. Figure 1 shows an overview of the proposed approach.

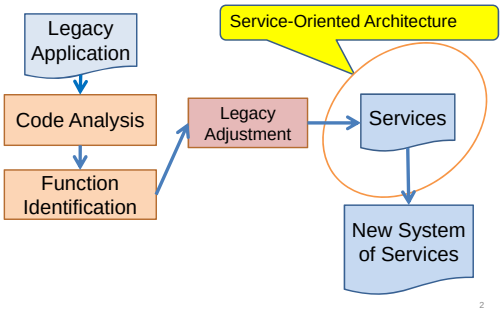

Figure 1: Overview of Proposed Approach

As the first step, whole legacy source code should be analyzed. And then the function identification step follows. In this step all the functions in source code should be identified and then they should be categorized. All the methods proposed in this framework are used based on this function identification and categorization. The proposed method converts almost all the functions in legacy source codes into relevant coding.

4. Proposed Framework

This framework addresses problems occurs when migrating legacy functions which are tightly coupled with GUI. These types of functions are very common in legacy applications which were developed in event driven programming languages. Process of this framework uses the source code of the legacy application and all the functions in this source code are used to create Web services or client side scripting functions. Even though, this framework uses the source codes, it does not change any single line of legacy functions. It just copies the functions into respective classes. This framework consists of a set of methods and these methods converts legacy functions into Web services or client side scripting functions.

4.1. Classification of functions

We introduced three types of function classifications. These classifications will be used in migration process of functions to services.

4.1.1. First Level of Classification

At the first level, functions in a source code are classified into three categories, as Fully Bound, Partially Bound, and Unbound. And this is done based on their degree of dependency on the GUI.

4.1.1.1. Fully Bound Function

If the internal working of a function is fully coupled with GUI, then it is categorized as fully bound function. Figure 2 shows an example for fully bound function.

```
Private Sub resetEmp()
  txtEmployeeId = ""
End Sub
```

Figure 2 Fully Bound Functions

4.1.1.2. Partially Bound Function

Functions which are coupled with GUI called as partially bound functions. Figure 3 shows an example for partially bound functions.

```
Private Function Uname() As String
  Dim rst As New ADODB.Recordset
  Dim sqlstr As String
  sqlstr = "select initials,Lname from emphed where empid='"
  &txtEmployeeID.Text& "'"
  rst.Opensqlstr, secuCon
  If rst.BOF = False Then
    Uname = rst!initials& " " &rst!lname
  Else
    Uname = ""
  End If
  rst.Close
End Function
```

Figure 3 Partially Bound Function

4.1.1.3. Unbound functions

Functions which are not coupled with GUI called Unbound Functions. Figure 4 is an example for unbound function.

```
Private Function Uname(empid As String) As String
  Dim rst As New ADODB.Recordset
  Dim sqlstr As String
  sqlstr = "select initials,Lname from emphed where empid='" &
  empid&  "'"
  rst.Opensqlstr, secuCon
  If rst.BOF = False Then
    Uname = rst!initials& " " &rst!lname
  Else
    Uname = ""
  End If
  rst.Close
End Function
```

Figure 4 Unbound Function

4.1.2. Second Level of Classification

In the second level of classification, function to function coupling is considered. Fully bound functions in the first level classification are not applicable in this classification since they are purely working on GUI.

4.1.3. Third Level of Classification

The objective of this classification is to identify the event handlers. This identification is used when developing the cloud version of legacy application.

5. Converting Functions into Services

After the classification of the legacy functions, they are converted to services or client side scripting functions. There are several methods proposed in this framework based on the above classification in order to migrate legacy functions into services.

5.1. Fully Bound Function to Service (FB to Service)

The mechanism to migrate the internal working of this fully bound functions to Web pages is named "***FB to Service***". In the FB to Service, internal working is moved to client side of the Web page and these workings are done by client side scripting coding.

5.1.1. Reestablishing Coupling

In FB to Service method, functionalities of fully bound functions will be shifted to client-side of the Web page. Hence, coupling of these functions with other functions or events has to be reestablished in different way. Under the FB to Service, this framework proposes a mechanism to address this issue. As the solution for this issue, a special class "toClientFunc" is provided in this proposal and it is encapsulated with Dynamic-Linked library (DLL) "webCom". The usage of the DLL is to establish coupling of dependency between components or functions

5.2. Partially Bound Functions to Service (PBF to Service)

The second proposed mechanism is called "***PBF to Service***". PBF to Service is for the partially bound functions which is the second category of the first level classification.

There are three types of partially bound functions. This classification is made based on their coupling levels.

1. Simple partially bound
2. Extended partially bound
3. Special Partially bound

5.2.1. Simple Partially Bound Function (SPBF)

A function is considered as a simple partially bound, if it is coupled with GUI by sharing values of objects on the user interface or they can be coupled with another fully bound function.

5.2.2. Extended Partially Bound Function (ExPBF)

Extended partially bound is an extended version of simple partially bound function. Functions belong to this category are coupled with another partially bound functions as well.

5.2.3. Special Partially Bound Function (SpPBF)

In the middle of the process of Special partially bound functions, they request some input from user for their further processing.

5.2.4. Technical Overview of *PB to Service*

PBF to Service of this proposed frame work is on providing mechanism for migration of partially bound functions to services. As explained in the above section, there are three kinds of partially bound functions. They need three different mechanisms for their migrations into services. In PBF to Service, three mechanisms have been proposed for each of these partially bound functions. In all the methods of PB to Service Scheme, each and every legacy function except fully bound is wrapped with an interface called "Gateway Method". Figure 5 shows an example for gateway Method.

5.2.4.1. Simple partially bound functions into service (SPB to Service)

This is the first method in PBF to Service, which converts simple partially bound functions (SPBF) into services. The key concept behind this method is the objects on the GUI are replaced by shared variables. In this framework classes are created for each and every form of the legacy application. Methods in these classes are the functions of their respective forms. Hence simple partially bound functions are now in the form of methods. Therefore they can be referred as "*Simple partially bound methods*".

```
Public Function G_getName (id As String) As String
    txtMemberID.text = id
    G-getName = getName
End Function
```

Figure 5 Gateway Method

5.2.4.2. Extended partially bound functions into service (ExPB to Service)

Under the PBF to Service, "***ExPB to Service***" is a proposed method for migration of extended partially bound functions to service. Same as the "*SPB to Service*" method, "*ExPB to Service*" also start from classes. The gateway method of "*ExPB to Service*" method has to do adjustments for its legacy function as in SPB to Service and additionally for its dependency function too.

5.2.4.3. Special partially bound function into Service (*SpPB to Service*)

Method "***SpPB to Service"*** is the proposed method to migration of special partially bound functions into service. After migrating the functions to service that is in an environment where GUI and business logic processing are done on two different and separate places, their original behavior won't be applicable. "*SpPB to Service"* method addresses this issue. "*SpPB to Service"* method consists of two

sub methods “SpPB Type-1 to Service” and “SpPB Type-2 to Service”

5.2.4.3.1. SpPB Type-1 to Service Method

“SpPB Type-1 to Service” method is same as to gateway function. “SpPB Type-1 to Service” method creates a gateway method as in other methods. Two tasks that is initialization of properties and methods call take place in this gateway method too. This method is the same for the type-1 and type-2. After the gateway method call, an additional method call “deviation method” is created. The name and parameters of this method should be same as name of the function or the method used in this legacy functions to get the user input during its execution. For example, “MsgBox” function in Visual Basic can be used in a method as follows.

“MsgBox("Are You Sure to pay 400 to the charity fund", vbYesNo, "charity Payment") = vbYes “

Then the new deviation method should be created as follows.

Public Function MsgBox(s1 As String, s2 As String, s3 As String) As Integer

The objective of this deviating method is to deviates the functional call made to MsgBox function in Visual Basic to this deviating method. Internal working of this deviating method provides required data for the rest of processing of the legacy function.

5.2.4.3.2. SpPB Type-2 to Service Method

The proposed method for convert Special partially bound function type 2 into Web service is referred as “***SpPB Type2 to Service”***. “*SpPB Type2 to Service”* Method is almost same as “*SpPB Type1 to Service”* method upto creation of deviating method. Deviating method in “*SpPB Type2 to Service”* sends a request to user who is at the client side for necessary data to be supplied. These data are then be used in main SpPB Type2 function.

5.3. Unbound Function to Service (UB to Service)

Converting unbound functions into services are rather simple and straight forward than partially bound functions. Since these are not coupled with GUI, there is no need for gateway function. These functions are copied into its related class directly without doing any additional work. Then this class is encapsulated in to related DLL.

6. Evaluation and Discussions

The framework presented in this paper is to be used to facilitate the migration process of legacy applications into cloud computing. Since this framework uses sauce code, high quality migration can be expected. Even though, these methods deal with source code of the legacy applications, they never touch the coding of functions except for the fully bound functions. In fact, they add an interface for each and every function of the source code. Hence, the internal working of legacy function will not be changed. Therefore, original performance can be preserved.

7. Future Works

As future studies, some research should be done to improve the speed of the new application. This framework is more suitable for small applications. However, to expand this to large-scale applications, there should be tool supporting. Hence find or developing suitable tool to automate this process is one of the main tasks to be done. Also, application of this framework to multiple languages is another quality to be added to this framework.

8. Conclusions

This paper proposes a framework to facilitates the migration process of legacy applications into cloud computing. This proposed framework addresses the problem related with coupling function with GUI. To address this problem, it proposes an interface for function called “Gateway Method”. Any request to original function should be made via this gateway method. This gateway method does some adjustment so that the legacy function is isolated or decoupled from GUI and can be accessible independently.

References

[1] G. Canfora, et al., Migrating Interactive Legacy Systems to Web Services, Proc. CSMR 2006. Mar. 2006, pp. 320-329.

[2] H. M. Sneed, et al., Creating Web Services from Legacy Host Programs, Proc. WSE’03, Sep. 2003, pp. 59-65.

[3] E. Stroulia, et al., From Legacy to Web through Interaction Modeling, Proc. ICSM’02, Oct. 2002, pp.320-329.

[4] T. Schulze, et al., Towards Providing Lightweight Access to Legacy Applications as Cloud-Based Services, Proc. ACIS 2010, Dec. 2010, http://aisel.aisnet.org/acis2010/47.