Technical Analysis

Vertex Financial Services – Supervisor

Presented by ***ArtinSoft Consulting Services***

Version 1.0

31/1/2008

Legal Notice

The information contained in this document represents the current view of ArtinSoft on the issues discussed as of the date of publication and is subject to change at any time without notice. This document and its contents are provided AS IS without warranty of any kind, and should not be interpreted as an offer or commitment on the part of ArtinSoft. This is a confidential document, and it is intended for review by Vertex Financial Services exclusively.

All company and product names mentioned herein may be trademarks or registered trademarks of their respective owners.

Contents

1 Introduction

This document presents an analysis of *Supervisor*, a Vertex Financial Services application written in Visual Basic 6.0, in order to assess the eventual migration of this program through the use of automation-assisted migration software and services provided by ArtinSoft.

ArtinSoft performed a five-day analysis with the following objectives:

- To diagnose the current state of the system towards an eventual migration to the C# .NET platform.
- To propose migration alternatives.

This analysis took place at the Vertex Financial Services offices in Cheltenham, England and at ArtinSoft's offices in San José, Costa Rica. One ArtinSoft consultant and several members of the Vertex Financial Services development team were involved in the process.

This is a confidential document, and it is intended for review by Vertex Financial Services exclusively.

2 Executive Summary

Supervisor is an in-house developed system written in Visual Basic 6.0. It is a stand-alone desktop application used for the configuration and monitoring of the Omiga application. The application is executed on Windows, and connects to both the Omiga application directly and with the back end running on SQL Server 2005 as the database.

The project outlined in this document details the strategy suggested to move the Supervisor application to C# and the .NET Framework using automated migration tools and services by ArtinSoft.

The total elapsed time to perform the migration activities is estimated at approximately four months, plus an optional additional month for on-site support and Additional Database Concurrency testing.

Two different migration options are provided in this document. The first of them is based on the assumption that Functional Equivalence will be reached using the SQLClient ADO.NET data provider. For the second option, it is assumed that Functional Equivalence will be achieved using the System.Data.Common classes and ArtinSoft helper classes. The two options can be performed using the same cost and duration estimates provided in the document.

Please refer to section 7.2 Project Calendar for more information on the duration of the project.

The offering presented by ArtinSoft includes:

- A software license for the Visual Basic Upgrade Companion (VBUC) at a cost of GBP 8,222.00
- Migration to Functional Equivalence at a cost of GBP 106,916.00
- An offer to provide support for Additional Database Concurrency Testing at a rate of GBP 70.00 per hour.

The overall migration budget, including VBUC license, as well as the migration to Functional equivalence is **GBP 115,138.**

3 Restrictions and Assumptions

1. This document is based upon interviews with Vertex Financial Services personnel involved in the development and maintenance of the applications. It

is therefore assumed that the information gathered for this analysis is accurate and reflects the actual status of the applications.

2. Effort and cost estimates provided in this document are based on the Visual Basic 6.0 code delivered to ArtinSoft during the week of January 14th, 2008. Due to expected source code changes, an adjustment to these estimates will be necessary prior to the beginning of the migration project

3. There are two migration options presented in this proposal:

   3.1. The first option assumes that the migrated application will reach Functional Equivalence using the ADO.NET SQL Client data provider as a replacement for ADO.

   3.2. The second option assumes that the migrated application will reach Functional Equivalence using classes from the System.Data.Common namespace and ArtinSoft Helper Classes to replace ADO database technology.

   Going with either of these options, however, does not vary the effort required to perform the migration.

4. This proposal does not consider additional Vertex requirements for the customization of the migration tool. Should these requirements arise in the future, they can be incorporated into the proposal by a modification to the scope of the project

5. This project has a dependency on the Omiga application. However, the migration project for the Supervisor application is regarded as an independent effort of the migration of Omiga to .NET.

6. This proposal assumes that there is no *Migration Cookbook* detailing the specific transformations expected by Vertex. If a specific migration pattern is

required, it can be handled through a Change Request before starting the project. ArtinSoft needs to evaluate the impact that any new requirement may have on the migration project.

7. The migrated version of Supervisor interacts with Omiga by means of direct manipulation of data in the database and by sending XML code to ASP pages through HTTP POST calls. This proposal assumes that the Migrated Version of Omiga will still retain the same database structure and ASP pages, so there will be no need to modify the Supervisor code to interact with Omiga.

4 Glossary of Terms

- Acceptance Criteria: A set of criteria that once validated in the Migrated Application deem it as acceptable.
- Change Request: A formal procedure used to modify the terms of an agreement
- DBMS: Database Management System
- Executable Application: Set of files or programs which can be executed independently and contain entry points to the application. The Migrated Application contains several Executable Applications.
- EWI: Errors, Warnings or Issues. These are comments added by the Migration Tool into the Migrated Application to indicate the developers that manual work may be needed or to explain a given conversion.
- Functional Equivalence: Refers to equivalent behavior between the Source Application and the Migrated Application, which is verified by having the Migrated Application produce the same output on the same Test Cases as the Source Application.
- Green Code: The output from the execution of the Migration Tools in the Target Language. No manual changes have been performed to make the code compile and run.

- Helper Classes: Helper Classes are classes coded in the target programming language which encapsulate certain functionality of a component or technology not directly supported by the target platform.
- LOC: Lines of Code, not including comments and blank lines.
- Migrated Application: The application written in C# and the .NET Framework programming language.
- Performance Testing: Evaluation of the performance of the application so that it meets certain performance metrics.
- Source Application: The application written in Visual Basic 6.0 programming language.
- UI: User Interface.
- Test Cases: Formal examinations designed to validate certain functionality
- Third-party Components: Any library or Application Programming Interface (API) not natively included in Visual Basic 6.0 or C# .NET.
- Time and Materials: Project execution schema in which ArtinSoft carries out activities defined by Vertex to achieve a given goal and where the consulting cost and materials required will be directly charged to Vertex according to an agreed hourly rate.
- VBUC: Visual Basic Upgrade Companion

5 Technical Analysis

5.1 Methodology

The analysis methodology for each of the applications was divided into three phases:

Design assessment: This phase evaluated the current application design to determine the advantages and disadvantages it presents towards the migration process. The following aspects were taken into account:

- N-Tier architecture, if any

- Server-side business logic
- Data repositories
- Web application, if any

Code assessment: This phase evaluated code size (in terms of Lines of Code) and the complexity as it would affect a migration process. The following aspects were taken into account:

- Code size
- Code complexity
- Code type (Visual Basic 6.0, C, C# and the .NET Framework, scripts, etc.)
- Presence of unsupported features

Migration strategy: This phase consists of an analysis of the current architecture, to determine a starting point for the migration process and the steps to follow with the intention to achieve 100% Functional Equivalence in every module.

5.2 Application Background

Supervisor is used by Omiga[1] administrators to manage the application; it is a stand-alone application run on the administrator's own desktop. It connects directly to the Omiga database, and also communicates with the Omiga components by making HTTP calls to Omiga*'s* ASP pages.

5.3 Application Structure, Operation and Deployment

The application consists of five Visual Basic projects (.vbp files):

1. **MSGOCX**
2. **OSGInterfaces**
3. **Security**

[1] For a description of the Omiga project and its migration requirements, refer to the Omiga Ready Document: *Vertex Ready Document 2.4.pdf*

4. **SecurityMgt**
5. **Supervisor**

The application is deployed using an *InstallShield* installer, and is used by Omiga administrators to manage the system. Any modifications to the installer for it to work with the Migrated Application are out of the scope of this proposal.

5.4 Support, Development and Maintenance

The support, development and maintenance tasks are performed by Vertex Financial Services at their offices in Cheltenham, U.K.

5.5 Testing Process and Structure

Currently, there are no written Test Cases for the Supervisor application. For the migration project, however, Vertex's testing team will develop the manual test cases necessary for the Acceptance Criteria of the migration project.

Vertex may also create automated test cases using Compuware Test Partner, as an addition to the Manual Test Cases. This proposal assumes that Vertex will only provide Manual Test Cases for the purpose of testing the Migrated Application.

5.5.1 Testing Plan

ArtinSoft will apply its formal testing process to the Migrated Application. This testing process follows the common and standard procedures accepted in software development. However, testing activities in a migration process have some differences when compared to other software development projects. In a migration project, there is a moment in which different functionalities from different application areas can be released. In this context, these areas are tested as soon as they are released, and any difference with the original application is reported as an error.

The main goal is to reach Functional Equivalence. In general, according to ArtinSoft experience, around 4 or 5 rounds of application testing are necessary the application to reach stability. ArtinSoft uses Test Cases provided by the customer to learn about the application and direct the testing process. Each test case is given a status after it is executed.

The testing process for a migrated application uses the industry's standard Testing Practices. Bugs are handled using a bug tracking system, which includes a verification process from QA every time a bug is set as "fixed" by a developer.

The Acceptance Criteria for the project will require having all the Test Cases in "pass" status and all bugs resolved and verified. Effort and duration estimates for testing activities are based on historical data, which includes programming technologies, amount of code lines and amount of forms, controls and other interfaces of the application.

5.5.2 Testing Assumptions

The testing process considers the following assumptions:

1. Vertex will provide a set of well documented and verified test cases, which will constitute the Acceptance Criteria of the project.
2. Vertex will provide any necessary help or advice required by ArtinSoft, in order to address doubts regarding the application being migrated.
3. All test cases can be successfully executed in the original application, producing the expected results.
4. ArtinSoft will have a working installation of the original application (Supervisor) that will be used for comparing the results of the test cases between the original and converted application. This makes it necessary to have a working Omiga installation in ArtinSoft's premises.

5.6 Design Assessment

This section elaborates on the architecture and coding practices of the current Visual Basic 6.0 application. The objective is to provide background information for the migration project.

The Original Application was written using Visual Basic 6.0 code. It is a stand-alone desktop application used for the configuration and monitoring of the Omiga application. The interaction between Supervisor and Omiga is accomplished in two ways:

- Direct data manipulation on the database: The Supervisor application and Omiga share the same database. Because of this, any updates on the database tables done by Supervisor is picked up by Omiga the next time it executes a query.
- POST XML code to Omiga's ASP pages: Omiga's business components, which hold all the business logic of the application, can be accessed through ASP web pages. Supervisor uses a POST HTTP event to send XML code to these ASP pages. The ASP pages then interpret the XML code, and call the corresponding methods on the business components., Supervisor interacts with the following Omiga web pages:
    - LaunchBatch.asp
    - DeleteCustomerLock.asp
    - UnlockApplicationAndCustomers.asp
    - LockCustomersForApplication.asp
    - GetBankDetails.asp
    - ValidateUserLogon.asp

It is important to mention that the migrated version of the Omiga application will still use ASP pages. This means that there will be no need to modify the Supervisor code to interact with new web architecture on Omiga.

The following diagram illustrates the interaction between Supervisor and Omiga:

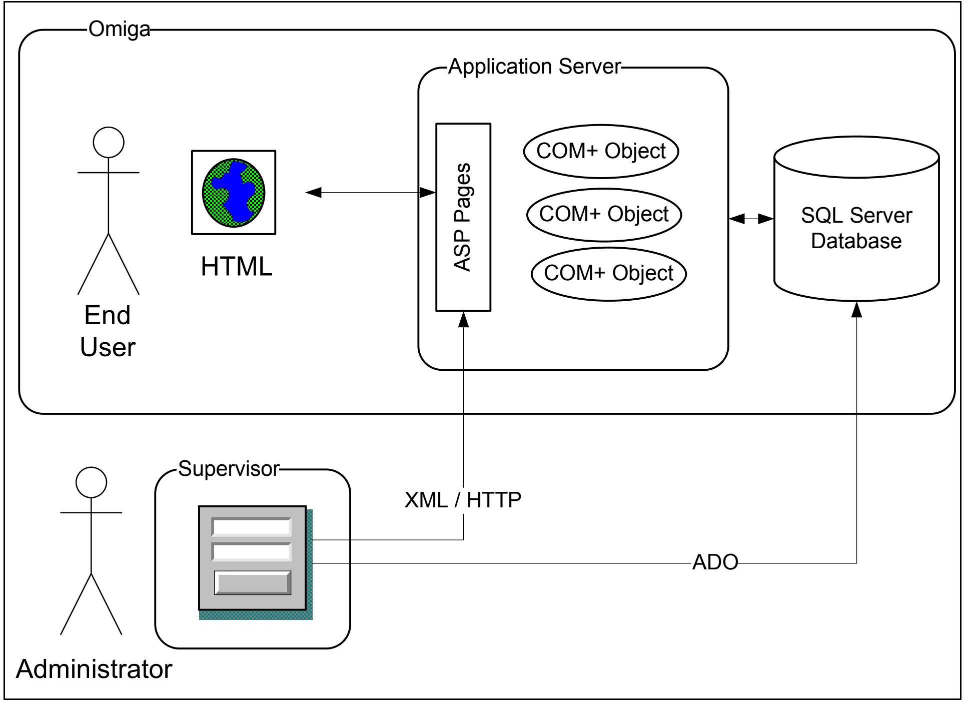


Figure 1. Supervisor / Omiga interaction

5.6.1 Application Architecture

As previously mentioned, the Supervisor application consists of five projects. The main project is called ***Supervisor.vbp***, and holds all the forms and business logic of the application. The ***MGSOCX.vbp*** project holds the User Controls used by the Supervisor application.

In addition to the MGSOCX controls, the application relies on a series of controls to provide certain functionality. These controls are:

- ADODB
- MSComctlLib
- MSComDlg
- MSMask
- MSXML2
- Scripting
- Stdole
- TabDlg

5.6.2 Current Environment

• Operating System:	Windows 2000 and XP
• Database:	SQL Server 2000 and 2005
• Visual Basic Version:	Visual Basic 6.0 SP 5

5.7 Code Assessment

5.7.1 Application Size and Distribution

The following table shows the application size and source code distribution, in terms of Lines of Code, for the complete migration project:

Table 1. Application Lines of Code

Programming Language	Lines of Code
Supervisor	
Visual Basic 6.0 Code	
Visual Lines:	42,583
Code Lines:	121,868
Blank Lines:	37,281
Comment Lines:	29,345
Total Lines:	**231,077**
Total EFECTIVE Lines (Visual + Code)	
Total Lines:	**164,451**

The following table details the source code distribution according to the different projects included in the application:

Table 2. VB6 Lines of Code per project

Project	Effective Lines of Code (VB6 Only)
MSGOCX	6,933
OSGInterfaces	150
Security	315
SecurityMgt	32
Supervisor	157,021
Total:	**214,574**

Notes:

- Blank lines and comments are not included in the Effective Lines of Code (ELOC) count.
- Information was taken from the results of ArtinSoft Code Analysis tools.

5.7.2 Modularity

The Supervisor migration consists of 5 (five) Visual Basic 6.0 projects. The *Supervisor.vbp* project, however, concentrates most of the lines of code. The following figure shows the distribution of lines of code between the different projects that make up the application:

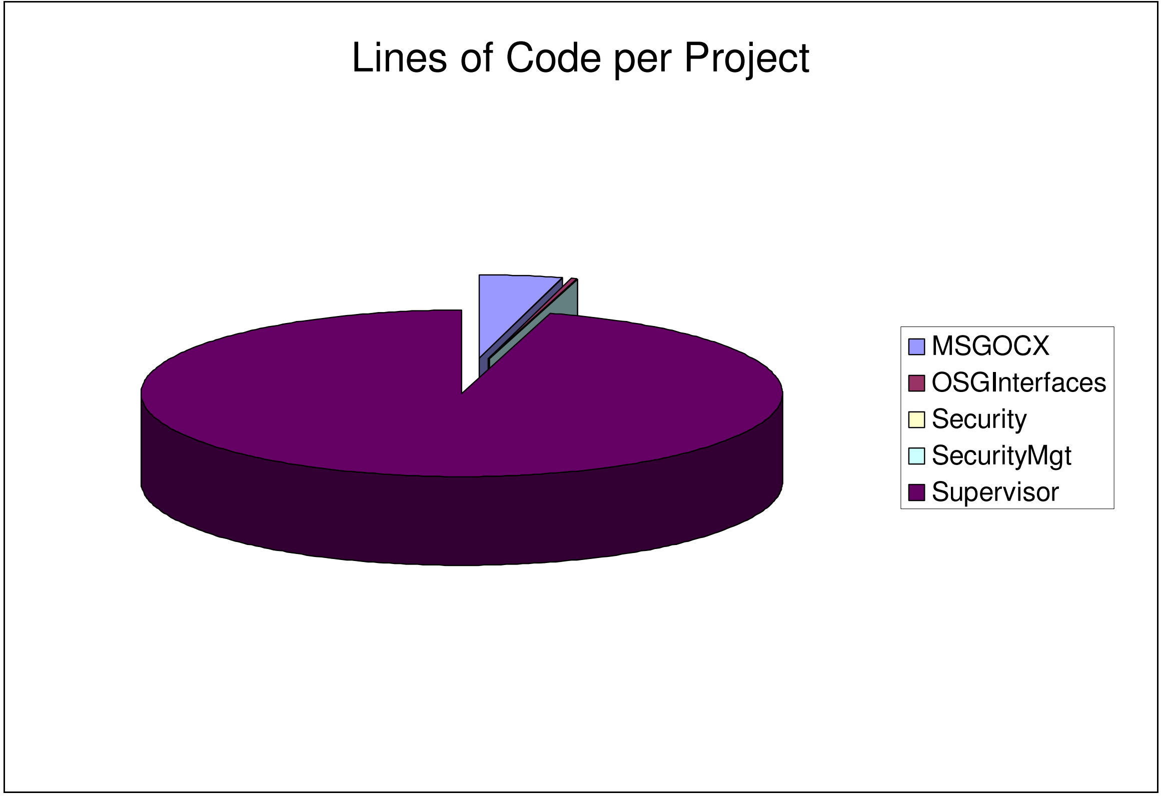


Figure 2. Lines of Code per Project

5.7.3 Complexity Guidelines

This section describes the complexity of the project for an eventual migration process. The following guidelines are used to categorize the application as Low, Medium, High or Very High Complexity:

- **Low**: no language structures incompatibilities found
- **Medium**: incompatibilities that can be upgraded with relatively simple manual changes were found
- **High**: application characteristics require a major re-architecture in order to migrate
- **Very High**: application requires migration tool configuration and/or extra parsing tools besides the re-architecture

Based on the results of the analysis performed on the Visual Basic 6.0 code, the migration to C# and the .NET Framework is considered to have a **Medium** complexity. The factors taken into account to come to this conclusion are:

- The data access of the application will be converted from ADO to ADO.NET using either the SqlClient data provider or classes from the *Sql.Data.Common* .NET namespace. These changes need to be tested and may cause complications that are not known at the time of the estimate.
- There are components that will be converted manually to their equivalent .NET versions, including:
  - MSDataGridLib
  - MSDataListLib
  - stdole.Font
  - MSXML2.ServerXMLHTTP

  Even though ArtinSoft analyzes the conversion of each one of them before presenting an estimate, there may be complications that are not known when planning the project.
- Most of the components used in the application are currently supported by VBUC for conversion to their .NET equivalents, including:
  - MSComctlLib
  - MSComDlg
  - MSMask
  - MSXML2
  - Scripting
  - Stdole
  - TabDlg
- No significant migration problems were identified on the Visual Basic 6.0 code.

6 Migration Project

This section describes the overall migration project for the Visual Basic 6.0 applications. This conversion will be performed following best practices and guidelines developed by ArtinSoft for migrations to C# and the .NET Framework.

6.1 Project Goals

Based on meetings held during the Ready on-site part of the assessment, Vertex Financial Services has the following objective for this migration project:

Table 3. Migration Project Goals

Goal	Solution
Move away from an unsupported platform.	Microsoft has announced the termination of support for Visual Basic 6.0 for March, 2008. Converting the application to C# moves Vertex Financial Services away from a legacy development environment and into a fully supported platform moving forward.

6.2 Proposed Target Environment

The migrated application will run in the following environment:

• Database:	SQL Server 2005
• Operating System:	MS Windows 2000, XP and Vista
• .NET Framework:	2.0

6.3 Migration Requirements

In addition to the migration of Supervisor source code from Visual Basic 6.0 to C# and the .NET Framework, Vertex Financial Services has provided the following requirements:

6.3.1 Migration of COM Components to .NET Native components

The application uses eight controls that relate to the user interface. One of the requirements for the project would be to remove these controls and replace them with a .NET equivalent. This section will detail the components that are currently converted automatically by the migration tool, and the components that need to be replaced manually after the automatic migration.

6.3.1.1 Components Supported by the VBUC

The Visual Basic Upgrade Companion already supports the conversion of the following components:

Component	Target .NET Component
MSComctlLib	
MSComctlLib.ColumnHeader	System.Windows.Forms.ColumnHeader
MSComctlLib.ImageList	System.Windows.Forms.ImageList
MSComctlLib.ListItem	System.Windows.Forms.ListViewItem
MSComctlLib.ListItems	System.Windows.Forms.ListView.ListViewItemCollection
MSComctlLib.ListSortOrderConstants	System.Windows.Forms.SortOrder
MSComctlLib.ListSubItem	System.Windows.Forms.ListViewSubItem
MSComctlLib.ListView	System.Windows.Forms.ListView
MSComctlLib.Node	System.Windows.Forms.TreeNode
MSComctlLib.StatusBar	System.Windows.Forms.StatusStrip
MSComctlLib.TabStrip	System.Windows.Forms.TabControl
MSComctlLib.TreeView	System.Windows.Forms.TreeView
MSComDlg	
MSComDlg.CommonDialog	ColorDialog, FontDialog, OpenDialog, SaveDialog or PrintDialog, depending on usage.
MSMask	
MSMask.MaskEdBox	System.Windows.Forms.MaskedTextBox
MSXML2	
MSXML2.DOMDocument	System.Xml.XmlDocument
MSXML2.DOMDocument40	System.Xml.XmlDocument

MSXML2.FreeThreadedDOMDocument	System.Xml.XmlDocument
MSXML2.IXMLDOMElement	System.Xml.XmlElement
MSXML2.IXMLDOMNode	System.Xml.XmlNode
MSXML2.IXMLDOMNodeList	System.Xml.XmlNodeList
Scripting	
Scripting.FileSystemObject	System.IO Classes
Scripting.TextStream	System.IO.FileStream
TabDlg	
TabDlg.SSTab	System.Windows.Forms.TabControl
TabDlg.StyleConstants	System.Windows.Forms.TabAppearance

6.3.1.2 Components Not Supported by the VBUC

The strategy to migrate those components not supported by the Visual Basic Upgrade Companion will consist in a manual replacement, to be performed after the automatic migration, with equivalent components. This strategy is recommended because of the low number of occurrences of the components, which makes a customization to the VBUC not cost-effective. The following table lists the components that will be converted using this approach, and the recommended .NET target component:

Component	Target .NET Component
MSDataGridLib	
MSDataGridLib.Column	System.Windows.Forms.DataGridViewColumn
MSDataGridLib.DataGrid	System.Windows.Forms. DataGridView
MSDataListLib	
MSDataListLib.DataCombo	System.Windows.Forms. ComboBox
MSXML2	
MSXML2.ServerXMLHTTP	System.Net.HttpWebRequest and System.Net.HttpWebResponse

It is important to mention that some of the functionality of these controls will not be translated to their corresponding .NET equivalent. This is due to limitations on the native .NET controls because of their different design philosophies. A complete list of the methods that are not supported can be found in Appendix D.

6.3.2 ADO Conversion

Vertex is interested in converting the current ADO code into ADO.NET. Right now, the Visual Basic Upgrade Companion performs this conversion by migrating the VB6 ADO code into ADO.NET code using the SQL Server ADO.NET Client found in the *System.Data.SqlClient* namespace.

This solution, using the ADO.NET SQL Client, is currently being redesigned by ArtinSoft to achieve a higher migration percentage. The target for this new migration is the provider-independent classes on the System.Data.Common namespace, introduced with ADO.NET 2.0, and an ArtinSoft-provided helper class. Appendix A contains a more detailed description of this solution.

The Supervisor application only needs to connect with SQL Server, so there is no need to have provider independence through the System.Data.Common interfaces. This proposal, however, contains both migration options, using either the SQL Client provider or the System.Data.Common as migration targets.

6.3.3 String Replacement

Several places of the application still use the old name of the company, *Marlborough Stirling.* These strings need to be changed to *Vertex Financial Services*. This replacement will be part of the migration project.

6.3.4 Circular Dependencies

Two projects in the application, Security and SecurityMgtInterface, have a circular dependency between them. During the migration project, it will be necessary to merge both projects into one, as Visual Studio .NET is not able to work with circular references.

6.3.5 Supervisor – Omiga Interaction

As mentioned before, the Supervisor application interacts with Omiga through both ASP pages and the database. This communication needs to continue working once the migration of both Supervisor and Omiga is complete.

The migrated .NET version of the Omiga application will continue using ASP pages, with .NET COM components powering those pages. This means that Supervisor's interaction with Omiga's ASP pages will continue working after the migration of Omiga is completed, without making additional changes.

As for the database, the migrated .NET version of Omiga will continue using the same database schema, so it will not be necessary to modify Supervisor's database access code to adapt to a new data schema.

For the migration process, ArtinSoft will require a working Omiga installation to test the Supervisor application. If the project is performed at the same time as the Omiga migration project, then the Supervisor and Omiga migration projects will share the same Omiga installation for testing purposes. If not, Vertex will provide a working Omiga installation for the project.

6.4 Use of Visual Basic Upgrade Companion

The Visual Basic Upgrade Companion will be used to accelerate the migration. This tool allows the creation of new migration rules, or mappings, to increase automatic migration coverage according to the specific needs of Vertex Financial Services. Specifically, the following source code areas will be totally or partially covered by VBUC customization:

1. Code conversions
   1.1. Generate C# code
2. Convert the following controls or libraries found in the Supervisor code.
   2.1. The following conversions are covered by the current version of Visual Basic Upgrade Companion:
      2.1.1. Microsoft Windows Common Controls (*MSComctlLib.ColumnHeader, MSComctlLib.ImageList,*

MSComctlLib.ListItem ,MSComctlLib.ListItems, MSComctlLib.ListSortOrderConstants, MSComctlLib.ListSubItem, MSComctlLib.ListView, MSComctlLib.Node, MSComctlLib.StatusBar, MSComctlLib.TabStrip, MSComctlLib.TreeView) to System.Windows.Forms equivalent controls

2.1.2. *MSComDlg.CommonDialog* to System.Windows.Forms equivalent controls

2.1.3. *MSMask.MaskEdBox* to System.Windows.Forms equivalent controls

2.1.4. *Microsoft XML* (*MSXML2.DOMDocument, MSXML2.DOMDocument40, MSXML2.FreeThreadedDOMDocument, MSXML2.IXMLDOMElement, MSXML2.IXMLDOMNode, MSXML2.IXMLDOMNodeList*) to classes in the *System.XML* namespace

2.1.5. *Scripting* classes (*Scripting.FileSystemObject*, *Scripting.TextStream*) to .System.IO equivalent classes

2.1.6. *TabDlg* controls (*TabDlg.SSTab*, *TabDlg.StyleConstants*) to *System.Windows.Forms* equivalent controls.

2.2. The conversion of the data access of the application to either one of the following components:

2.2.1. *ADO* to ADO.NET using the interfaces from the *System.Data.Common* namespace and ArtinSoft helper classes

2.2.2. *ADO* to ADO.NET using the SQL Client data provider.

6.5 Pre-migration Tasks

Before handing over the Visual Basic 6.0 code for migration, ArtinSoft recommends that Vertex performs the following pre-migration tasks to speed up the migration project itself:

6.5.1 Code Advisor

In order to improve the percentage of automation for the project, ArtinSoft recommends the execution of the *Code Advisor for Visual Basic 6.0*[2] on the Visual Basic 6.0 source code, and to implement the changes suggested.

It is important to mention that many of the changes suggested by the Code Advisor are automatically performed by VBUC.[3]

6.5.2 Compilation Directives

The Supervisor code uses several conditional compilation directives. At migration time, VBUC will only convert the code inside those directives if the directive evaluates to True. Because of this, ArtinSoft recommends the elimination of these compilation directives from the source code by Vertex before starting the migration.

6.5.3 Code Cleanup

Some of the functionality of the application is no longer used. Vertex will review the application and remove all the unused code before starting the migration.

6.6 Database Concurrency Testing

Applications that are converted from Visual Basic 6.0 into C# and the .NET framework, due to the characteristics of both platforms, are not expected to be affected from significant performance issues. The Supervisor application, however, has database concurrency requirements that need to be satisfied by the migrated application as part of the Acceptance Criteria. These concurrency requirements are currently satisfied by the Original Application. Vertex will provide Test Cases that test the database concurrency of the application.

[2] http://www.microsoft.com/downloads/details.aspx?FamilyID=A656371A-B5C0-4D40-B015-0CAA02634FAE

[3] Visual Basic Upgrade Companion vs. Code Advisor, URL: http://www.artinsoft.com/VB-Upgrade-Companion-vs-CodeAdvisor.aspx

ArtinSoft will have access to these Test Cases at the beginning of the project, and will validate them against the Original Application.

In addition to the Test Cases mentioned in the previous paragraph, Vertex will perform Additional Database Concurrency Testing once the application meets the Acceptance Criteria and they receive the code from ArtinSoft. For this testing, ArtinSoft can assign resources to help out with any issue that may come up. This additional support during the Additional Database Concurrency Testing is offered on a Time and Materials basis.

6.7 Project Structure and Responsibilities

The following figure shows the stages of the migration process for Vertex Financial Services Supervisor application:

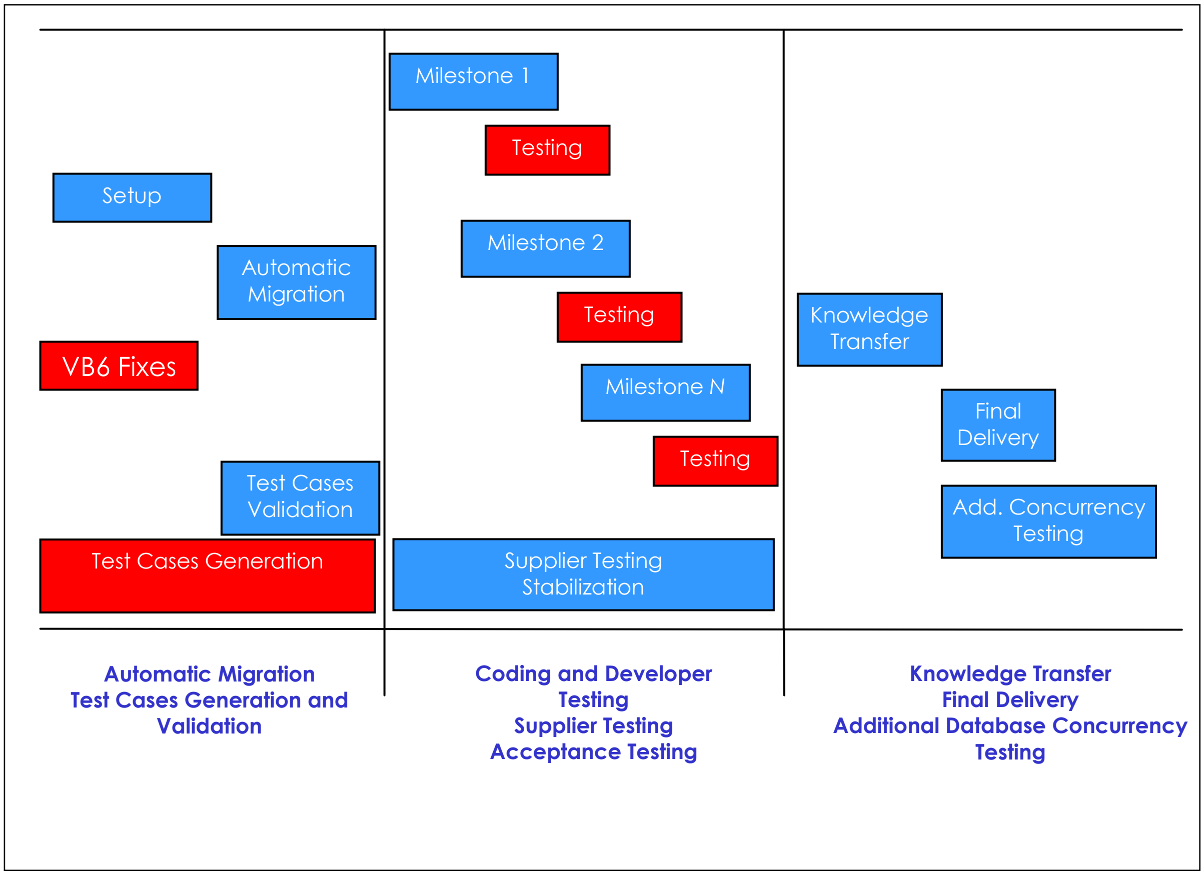


Figure 3. High-Level Project Layout (not to scale). The Blue boxes represent activities carried out by ArtinSoft. The Red boxes correspond to activities performed by Vertex Financial Services..

The preceding figure shows a general overview of the main phases that will form the migration project, for illustration purposes. It shows the order in which the different activities should be performed. The length of each activity does not represent its expected duration. These phases and activities are described below.

6.7.1 Setup

This stage covers the source code installation and project kick-off.

6.7.1.1 Kick-off

A kick-off conference call would take place with the ArtinSoft and Vertex Financial Services personnel involved in the project. The following topics will be covered in this initial project meeting:

- Project team organization
- Objectives
- Deliverables and deadlines
- Communication

6.7.1.2 Installation and Configuration

Machine installation and configuration; source code and components installation. Migration machines should duplicate a Vertex Financial Services development configuration.

It will also be necessary to install a fully working version of the Omiga application to verify the functionality of the application.

6.7.2 VB6 Fixes

Before delivering the code to ArtinSoft, it is recommended for Vertex to perform some Pre-migration Tasks on the VB6 source code. ArtinSoft's recommended tasks can be found in section 6.5 Pre-migration Tasks.

6.7.3 Automatic Migration

All Visual Basic 6.0 source code will be converted by the ArtinSoft Visual Basic Upgrade Companion to C# and the .NET Framework. Some specific details, including migration warnings and features that are not automatically migrated, will be manually reviewed and converted in subsequent phases of the project.

6.7.4 Test Cases Generation and Validation

ArtinSoft requires a mechanism to validate the functionality of the application to sign off on it as being "Functionally Equivalent". This is usually achieved through detailed Test Cases that can be used by the ArtinSoft testing team to validate the migration, and later by the Vertex Financial Services testing team to give the final acceptance. Vertex Financial Services must provide ArtinSoft with this detailed set of Test Cases before the Coding and Developer Testing stage begins. ArtinSoft will validate the Test Cases against the Source Application.

6.7.5 Coding and Developer Testing

Most of the manual work after the automatic migration process is done during this phase. The main tasks that will be performed during this phase are:

- **Manual fixes after automatic migration**: Resolve existing compilation errors in the C# code that may result from the migration process and resolve pending migration issues.
- **Runtime Bug Fixing**: During this phase, the developers involved in the migration process test the migrated Vertex application and solve issues found during testing. This testing will reduce the number of issues found during QA and will be organized according to system functionality. This testing would be performed using the Test Cases provided by Vertex Financial Services as a guide.

The same process is repeated for each of the milestones in the project. The detail on the milestones for the project is in section *6.9 Milestones*. After each milestone, Vertex will receive the corresponding source code to start the testing process as early as possible.

6.7.6 Supplier Testing and Stabilization

When the tasks performed during Coding and Developer Testing are completed, ArtinSoft's QA team will execute the manual Test Cases provided by Vertex Financial Services.

Testing will be scheduled according to the structure described in the Milestones section of this document to allow delivery of partial drops of the migrated application. These drops will allow Vertex Financial Services to start their testing process while ArtinSoft completes the migration of the remaining modules.

ArtinSoft may request advice from of Vertex Financial Services during this phase to correctly execute the Test Cases.

6.7.7 Acceptance Testing

Vertex Financial Services QA team will start the testing process with the first milestone during the Supplier Testing stage. During this process, the functionality of the application will be validated against the test cases developed by Vertex Financial Services, and ArtinSoft will provide fixes for any issues raised during this stage.

During this stage, ArtinSoft recommends having at least one consultant work alongside Vertex Financial Services testing team to provide support during the testing process and improve communication between the teams.

6.7.8 Knowledge Transfer

During this phase, ArtinSoft will train Vertex Financial Services technical personnel to understand the migrated code and maintain the application going forward. The following topics will be covered during this training:

- **C# and the .NET Framework Environment**: How to install and configure the migrated application.

- **Concept Mapping**: How the Visual Basic 6.0 concepts and instructions are mapped into the new platform.
- **Target Code Structure**: How the migrated code is structured in C# and the .NET Framework. For example, how the directory is structured on the new platform, how to handle global variables, records, etc.
- **Application Maintainability**: How to maintain, modify and add new features to the migrated application.
- **Application Enhancement**: How to enhance the migrated application in C# and the .NET Framework, how to improve the GUI, how to improve the performance in C# and the .NET Framework, among others.

This activity will be performed if requested by Vertex and is not considered in the pricing and time estimate presented in this proposal.

6.7.9 Final Delivery

ArtinSoft will provide Vertex Financial Services with the final delivery of the code along with detailed documentation. The documentation will include:

- Installation Guide
- Release Notes
- Bug Fixing log

This will be the last of the tasks included in the fixed price proposal for the migration of Supervisor.

6.7.10 Additional Database Concurrency Testing

Once the application is delivered and accepted as functionally equivalent, Vertex will perform Additional Database Concurrency testing at their offices with a more realistic environment. ArtinSoft can send one or more consultants to support Vertex efforts during this stage of the project. This activity will be performed on a Time and Materials basis.

6.8 Resource Allocation

Cost estimates are based on the assumption that ArtinSoft will provide most of the resources necessary to complete the project. Vertex Financial Services would be required to provide technical assistance during the migration and to develop the Test Cases what will serve as acceptance criteria. Regarding technical assistance, Vertex Financial Services needs to make available the following resources during the project:

- At least one main point of contact per area of the application
- A project manager
- Testing team resources to create the test cases and the testing methodology to be used to validate the application
- Testing team resources to provide guidance to ArtinSoft's testing team during the Supplier Testing stage when required by ArtinSoft.
- Testing resources to perform Acceptance Testing

6.9 Milestones

The project milestones will consist of the following:

Table 4. Project Milestones

Milestone	Detail
M0. Green Code	Green Code Delivery: All the VB6 code converted to C#
M1. Support Projects	Solve all EWIs and compilation errors on the MSGOCX.vbp, OSGInterfaces.vbp, Security.vbp and SecurityMgtprojects.vbp projects Remove the circular dependency on the Security and SecurityMgt projects by merging the SecurityMgt code into the Security project Perform Developer Testing on the MSGOCX

	user controls
M2. Supervisor Framework	Solve all EWIs and compilation errors on the files that comprise the framework used by the Supervisor screens. Migrate and test the Global Parameters screen
M3. Supervisor Screens 1	Migrate and test one half (approximately) of the remaining screens.
M4. Supervisor Screens 2	Migrate and test the remaining screens.

Due to resource constraints, Vertex is currently not able to provide the list of files that will be included in each specific milestone. The actual list of files that go into each milestone will be determined by Vertex before work on the Statement of Work is started. The actual timeframe of each component will be known at that time.

After milestones two (Supervisor Framework), three (Supervisor Screens 1) and four (Supervisor Screens 2), Vertex will receive the code to start with the testing process for the application.

7 Estimation

The purpose of this section is to provide an estimate of the time required to migrate the application. Estimates are calculated using ArtinSoft's Code Analysis tools combined with past experience on similar projects. Productivity for each portion was adjusted according to the size and complexity of the application.

The estimates provided in this section apply for two different approaches. The first of them is based on the assumption that Functional Equivalence will be reached using the ADO.NET SQL Client data provider. The second approach assumes that Functional Equivalence will be achieved using the interfaces from the System.Data.Common namespace and ArtinSoft's helper classes. There is no difference in effort or schedule to achieve Functional Equivalence with either option.

Based on the calculated productivity levels, time estimation for each stage detailed in the document is as follows.

7.1 Effort Estimate

Table 5.5 Effort Estimate – Visual Basic 6.0 Migration

Project Part	Responsible	Effort (person- days)
Project kick-off	ArtinSoft	3
Setup and automatic migration	ArtinSoft	15
Coding and developer testing	ArtinSoft	227
Test cases preparation and revision	Vertex Financial Services	N/A
Supplier Testing	ArtinSoft	41
Acceptance Testing Support & Bug Fixing	ArtinSoft	109
Acceptance Testing	Vertex Financial Services	N/A
Project Management	ArtinSoft	54
Total		**449 Person-days**

7.2 Project Calendar

The following effort estimate applies to either migration options, using the ADO.NET SQL Client data provider or using the interfaces from the System.Data.Common namespace and ArtinSoft's helper classes.

The preliminary dates for each of the major milestones are as follows:

Table 6. Project Calendar

Project Activities		
Activity	**Start Date**	**End Date**
Project kick-off	3/3/08	3/7/08
M0: Green Code	3/7/08	3/21/08

M1: Support Projects	3/24/08	N/A
M2: Supervisor Framework	N/A	N/A
M3: Supervisor Screens I	N/A	N/A
M4: Supervisor Screens 2	N/A	6/30/08
Migration Project Timeframe	**3/3/08**	**6/30/08**

The date for the milestones will be calculated once Vertex determines the specific code to be migrated at each milestone. This calendar applies to either migration options, using the ADO.NET SQL Client data provider or using the interfaces from the System.Data.Common namespace and ArtinSoft's helper classes.

These dates are preliminary and will be adjusted during the project kick-off. The project plan makes the following assumptions:

1. It would be possible to start the conversion of the application by the beginning of March, 2008.
2. The estimate and price may change once the final version of the source code is delivered to ArtinSoft.

8 On-site visits

ArtinSoft recommends several on-site visits from ArtinSoft consultants to the Vertex Financial Services premises during the migration project. This section elaborates on the visits required.

8.1.1 Acceptance Testing

ArtinSoft recommends having one consultant on-site for the duration of the testing stage. This has the following advantages:

- Improves the communication between the Vertex Financial Services and ArtinSoft teams

- Improves the response time for any decision that needs to be made by both parties
- Increases the involvement of the Vertex Financial Services engineering teams
- Allows further integration between the teams

8.1.2 Project Final Delivery and Acceptance

At the end of the project, one consultant from ArtinSoft would travel to the Vertex Financial Services premises to bring all project deliverables and provide any necessary knowledge transfer. The main objectives of this visit are:

- Knowledge transfer
- System deployment
- Final acceptance
- Project closing

The duration of this visit is estimated to be one week.

8.1.3 Additional Database Concurrency Testing (optional)

ArtinSoft recommends having one consultant on-site for the duration necessary to conduct the database concurrency tests. The consultant will provide support during the testing efforts, as well as guidance and expertise on the migrated application when issues arise. This additional support would be performed on a Time and Materials basis and it is not included in the effort and costs estimates shown in this document.

9 Anticipated Project Budget

The following budget reflects an approximation of the cost ArtinSoft expects would be incurred to perform the full migration of the Supervisor Visual Basic 6.0 code according to the criteria stated in this document. This is only an estimate, and should not be construed as a final offer to perform the migration at the stated price. A final fixed price proposal for the migration of the Supervisor source code will be created once the final code to be migrated is provided to ArtinSoft.

9.1 Project Cost

The following table provides an approximate cost for the Full Migration as explained in this document:

Table 7. Full-Migration estimated Budget

Option	Estimated Price
Migration Project (Functional Equivalence)	
Conversion of Supervisor to C# and the.NET Framework	GBP 106,916.00
Visual Basic Upgrade Companion License	GBP 8,222.00
Total Cost (Migration Project):	**GBP 115,138.00**

The prices do not include any travel and daily expenses.

9.2 Additional Database Concurrency Testing

Once the migration project is completed, ArtinSoft can provide Vertex Financial Services with additional support as needed in accordance with the prices stated in the following table:

Table 8. Support Budget

Option	Price
Cost per hour of Additional Database Concurrency Testing support	GBP 70.00

The prices do not include any travel and daily expenses.

10 Appendix A – ADO.NET System.Data.Common Migration

10.1 Description

Microsoft incorporated the System.Data.Common namespace in the.Net Framework 2.0, which contains classes, intended to be used as the base for all data provider implementation. This architectural decision allows application designers and programmers to use patterns that are data provider-agnostic on their data access layers.

VB6 offers several ways to access a database, all of them using COM libraries like ADO and DAO.

The most common structure used to retrieve data from the database in VB6 is the RecordSet. It is basically a collection of rows retrieved from the database using a specific SQL command.

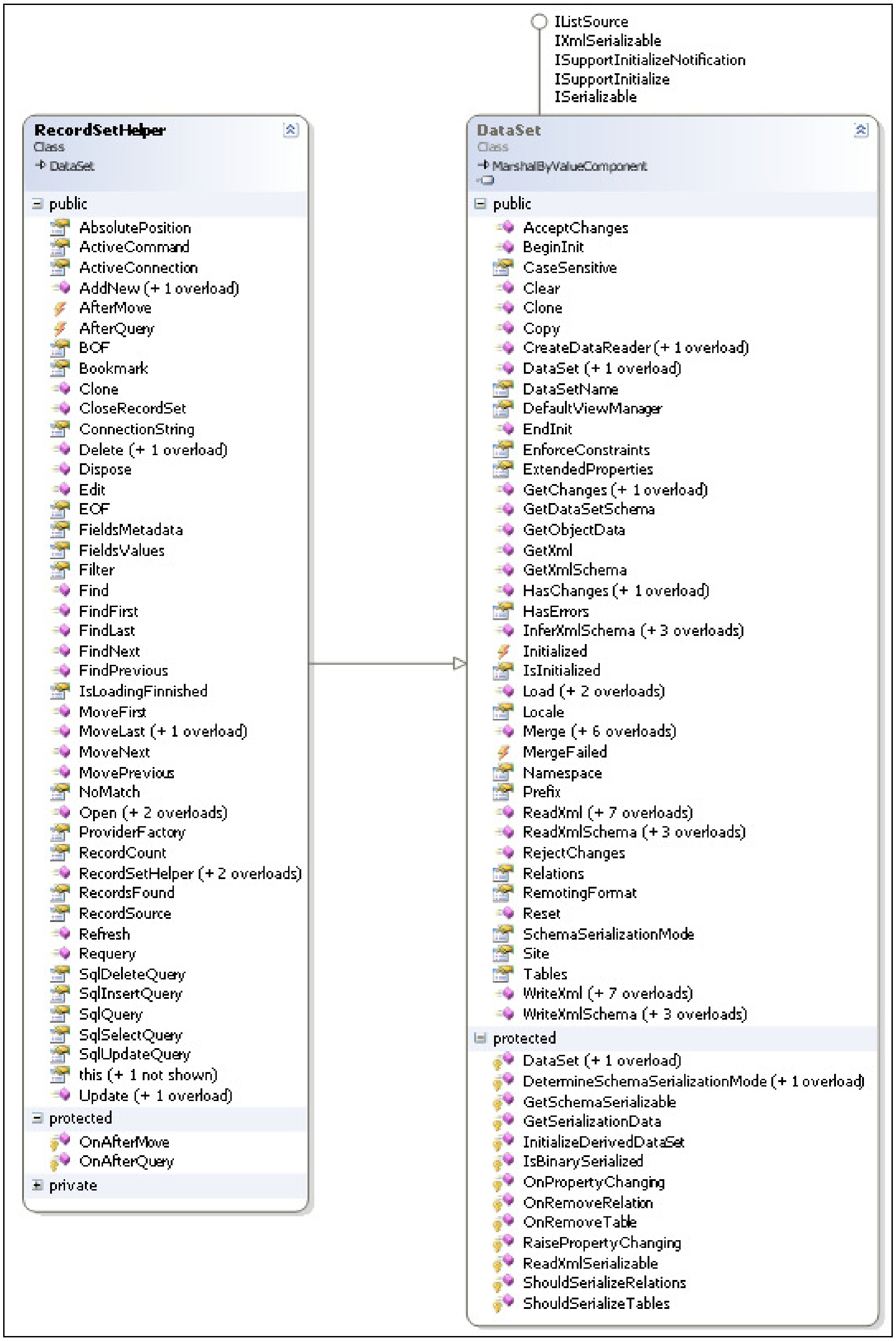
The .NET platform provides the DataSet components, which like the RecordSet, holds the data retrieved from the database.

There are several differences between those components. The most important difference in terms of Functional Equivalence is the capability of the Recordset to hold the current position and make all the operations on that record.

To accomplish the same functionality in C# .NET, ArtinSoft will use a Helper class to handle all RecordSet operations. Internally, this class will have all the necessary infrastructure to handle all database requests using DataAdapters, and using only classes from the System.Data.Common namespace.

10.2 Helper Class Design

The Helper class extends the DataSet class, giving it more functionality without breaking the natural .ADO.NET architecture.



The following code is part of the helper code. This example is for the *Open* methods that encapsulate the DataSet population logic.

```
#region Open Operations
        private void OpenRecordset()
        {
            operationFinished = false;
            DbDataAdapter dbAdapter = CreateAdapter(activeConnection);
            dbAdapter.Fill(this);
            operationFinished = true;
            currentView = Tables[0].DefaultView;
            currentView.AllowDelete = true;
            currentView.AllowEdit = true;
            currentView.AllowNew = true;
            if (Tables[0].Rows.Count == 0)
                index = -1;
            else
                MoveFirst();
            newRow = false;
            foundRecordsCounter = 0;
            OnAfterQuery();
        }

        public void Open()
        {
            if (activeConnection == null && activeCommand != null &&
activeCommand.Connection != null)
               ActiveConnection = activeCommand.Connection;
            else
                throw new InvalidOperationException("The ActiveConnection
property must be set before calling this method");
            OpenRecordset();
        }

        public void Open(DbCommand command, String connectionString)
        {
            this.connectionString = connectionString;
            Open(command, CreateConnection());
        }
        public void Open(DbCommand command, DbConnection connection)
        {
            ActiveConnection = connection;
            activeCommand = command;
            Open();
        }

        private void Open(String SQLstr, String connectionString)
        {
            this.connectionString = connectionString;
            CommandType commandType = getCommandType(SQLstr);
            DbCommand command = providerFactory.CreateCommand();
            command.CommandText = SQLstr;
            command.CommandType = commandType;
            Open(command, connectionString);
        }

        #endregion
```

10.3 Advantages

This approach has the following advantages:

1. It provides clearer and more readable code in C#. By using this approach there is no need to use generated temporary variables. The generated code will only have a Helper class call.
2. Since the Helper class inherits from the regular ADO.NET DataSet; it will be easy to integrate it with ADO.NET-compliant data access frameworks in the future. These frameworks will be able to use the Helper class as a regular DataSet without any limitations.
3. It minimizes the manual effort required on cases where the RecordSet is handled by more than one function or method with a position dependency.
4. The Helper class source code will be included in the deliverables of the project
5. It speeds up the migration project by minimizing manual work on the data access code after automatic migration.

10.4 Examples

The following are examples of the transformations needed with and without the Helper approach.

10.4.1 Opening a RecordSet

10.4.1.1 Source Code

```
Dim mRS As ADODB.Recordset
SQL = ConvertOracleToSqlServer(dbc, SQL)
mRS.Open SQL, dbc, adOpenForwardOnly, adLockReadOnly
```

10.4.1.2 Without Helper

```
DataSet mRS = new DataSet();
//We need a provider to create the dataadapter and command
//instances
DbProviderFactory dbProviderFactory =
DbProviderFactories.GetFactory("");
```

```
SQL = ConvertOracleToSqlServer(dbc, SQL);
//We need a provider to create a command instance
DbCommand selectcommand = dbProviderFactory.CreateCommand();
selectcommand.CommandType = CommandType.Text
selectcommand.CommandText = SQL
selectcommand.Connection = dbc
//An adapter instance is needed to fill the dataset
DbDataAdapter adapter = dbProviderFactory.CreateDataAdapter();
adapter.SelectCommand = selectcommand;
adapter.Fill(mRS, "table");
```

10.4.1.3 With Helper

```
RecordSetHelper mRS = new RecordSetHelper();
SQL = ConvertOracleToSqlServer(dbc, SQL);
//A direct call to the helper class which takes care of the
//adapters and command creations
mRS.Open(SQL, dbc);
```

10.4.2 Updating a RecordSet

10.4.2.1 Source Code

```
Set rsUpdate.ActiveConnection = dbc
rsUpdate.UpdateBatch
```

10.4.2.2 Without Helper

```
//We need a provider to create the dataadapter and command
//instances
DbProviderFactory dbProviderFactory =
DbProviderFactories.GetFactory("");
SQL = ConvertOracleToSqlServer(dbc, SQL);
```

```
//We need a provider to create a command instance
//The select command needs to be provided in order to infer the
//proper commands for insertion, deletion and updates
DbCommand selectcommand = dbProviderFactory.CreateCommand();
selectcommand.CommandType = CommandType.Text
selectcommand.CommandText = “Select”
selectcommand.Connection = dbc
//An adapter instance is needed to update the dataset
DbDataAdapter adapter = dbProviderFactory.CreateDataAdapter();
adapter.SelectCommand = selectcommand;
//We need a command builder to infer the proper sql commands
DbCommandBuilder                    cmdBuilder                    =
dbProviderFactory.CreateCommandBuilder();
cmdBuilder.DataAdapter = adapter;
adapter.UpdateCommand = cmdBuilder.GetUpdateCommand();
 adapter.InsertCommand = cmdBuilder.GetInsertCommand();
adapter.DeleteCommand = cmdBuilder.GetDeleteCommand();
adapter.Update(rsUpdate);
```

10.4.2.3 With Helper

```
rsUpdate.ActiveConnection = dbc;
rsUpdate.UpdateBatch();
```

10.4.3 RecordSet Iteration

10.4.3.1 Source Code

```
With prsChildRecordset
  .Filter = adFilterPendingRecords
  If Not .BOF And Not .EOF Then
    .MoveFirst
    Do Until .EOF
       .Fields(psKey).Value = pvValue
       .Update
       .MoveNext
 Loop
```

10.4.3.2 Without Helper

```
//Due to the filter property and value this has to be done
DataTable changedDt = prsChildRecordset.Tables[0].GetChanges();
foreach (DataRow row in changedDt.Rows)
{
   row[psKey] = pvValue;
   row.AcceptChanges();
}
```

10.4.3.3 With Helper (not optimized)

```
prsChildRecordset.Filter = adFilterPendingRecords;
if(!adFilterPendingRecords.BOF && !adFilterPendingRecords.EOF)
{
   adFilterPendingRecords.MoveFirst();
   do{
       adFilterPendingRecords[psKey] = pvValue;
       adFilterPendingRecords.Update();
       adFilterPendingRecords.MoveNext();
   }
   while(!adFilterPendingRecords.EOF)
}
```

10.4.3.4 With Helper (optimized)

```
//Due to the filter property and value this has to be done
DataTable changedDt = prsChildRecordset.GetChanges();
foreach (DataRow row in changedDt.Rows)
{
   row[psKey] = pvValue;
   row.AcceptChanges();
}
```

11 Appendix B – Project Statistics

Project	Total Files	Classes	Modules	UserControls	Designers	Forms
MSGOCX	17	3	5	9	0	0
OSGInterfaces	2	2	0	0	0	0
Security	4	3	1	0	0	0
SecurityMgt	1	1	0	0	0	0
Supervisor	435	311	11	0	0	113
Total	**459**	**320**	**17**	**9**	**0**	**113**

12 Appendix C – Recommended Upgrade Order

The following table details, per project, the recommended upgrade order, based on the dependencies and references of files within each project.

Project	File
MSGOCX	SUPERVISOR\MSGOCX\Classes\GeneralAssist.cls
	SUPERVISOR\MSGOCX\User Controls\MSGDataCombo.ctl
	SUPERVISOR\MSGOCX\User Controls\MSGListView.ctl
	SUPERVISOR\MSGOCX\Classes\RegistryAssist.cls
	SUPERVISOR\MSGOCX\Modules\SortListView.bas
	SUPERVISOR\MSGOCX\Modules\KeyPress.bas
	SUPERVISOR\MSGOCX\Modules\Constants.bas
	SUPERVISOR\MSGOCX\Modules\TabUtils.bas
	SUPERVISOR\MSGOCX\Modules\Utils.bas
	SUPERVISOR\MSGOCX\Classes\ErrorHandling.cls
	SUPERVISOR\MSGOCX\User Controls\MSGComboBox.ctl
	SUPERVISOR\MSGOCX\User Controls\MSGHorizontalSwapList.ctl
	SUPERVISOR\MSGOCX\User Controls\MSGMultiLine.ctl
	SUPERVISOR\MSGOCX\User Controls\MSGPasswordEditBox.ctl
	SUPERVISOR\MSGOCX\User Controls\MSGTextMulti.ctl
	SUPERVISOR\MSGOCX\User Controls\MSGEditBox.ctl
	SUPERVISOR\MSGOCX\User Controls\MSGDataGrid.ctl
OSGInterfaces	SUPERVISOR\OSG Interface\Classes\UserAdminDetails.cls
	SUPERVISOR\OSG Interface\Classes\SubSystem.cls
Security	SUPERVISOR\SecurityManager\Classes\SecurityResource.cls
	SUPERVISOR\SecurityManager\Modules\ADOHelper.bas
	SUPERVISOR\SecurityManager\Classes\SecurityResourceList.cls
	SUPERVISOR\SecurityManager\Classes\SecurityManager.cls
SecurityMgt	SUPERVISOR\SecurityManager\Classes\ISecurityManager.cls
Supervisor	SUPERVISOR\Super\Modules\TabUtils.bas

	SUPERVISOR\Super\Modules\errConstants.bas
	SUPERVISOR\Super\Modules\comment.bas
	SUPERVISOR\Super\Classes\RegistryAssist.cls
	SUPERVISOR\Super\Modules\EncryptAssist.bas
	SUPERVISOR\Super\Modules\StdData.bas
	SUPERVISOR\Super\Forms\frmMaintIntRates.frm
	SUPERVISOR\Super\Forms\frmSplash.frm
	SUPERVISOR\Super\Forms\frmTooltip.frm
	SUPERVISOR\Super\Classes\GuidAssist.cls
	SUPERVISOR\Super\Classes\SwapExtra.cls
	SUPERVISOR\Super\Classes\ObjectContext.cls
	SUPERVISOR\Super\Classes\ProgressBar.cls
	SUPERVISOR\Super\Classes\PopulateDetails.cls
	SUPERVISOR\Super\Classes\SupervisorUpdateDetails.cls
	SUPERVISOR\Super\Classes\Validation.cls
	SUPERVISOR\Super\Classes\ParserAssistSP.cls
	SUPERVISOR\Super\Classes\TreeItem.cls
	SUPERVISOR\Super\Classes\UserAccess.cls
	SUPERVISOR\Super\Classes\TreeViewCS.cls
	SUPERVISOR\Super\Classes\SecurityManagerCS.cls
	SUPERVISOR\Super\Classes\ListViewCS.cls
	SUPERVISOR\Super\Classes\DeleteObjectCS.cls
	SUPERVISOR\Super\Classes\PasswordData.cls
	SUPERVISOR\Super\Modules\ConvertAssist.bas
	SUPERVISOR\Super\Classes\MortProdNatureOfLoanCS.cls
	SUPERVISOR\Super\Classes\MortProdProdClassCS.cls
	SUPERVISOR\Super\Classes\MortProdIncomeStatusCS.cls
	SUPERVISOR\Super\Modules\Constants.bas
	SUPERVISOR\Super\Modules\MenuContants.bas
	SUPERVISOR\Super\Modules\Utils.bas
	SUPERVISOR\Super\Modules\Logging.bas
	SUPERVISOR\Super\Forms\frmDisplayError.frm
	SUPERVISOR\Super\Forms\frmProductGrouping.frm
	SUPERVISOR\Super\Forms\frmEditInterestRates.frm
	SUPERVISOR\Super\Forms\frmRegSettings.frm
	SUPERVISOR\Super\Classes\TreeData.cls

	SUPERVISOR\Super\Classes\PayProtRatesSetTable.cls
	SUPERVISOR\Super\Classes\FixedParametersTable.cls
	SUPERVISOR\Super\Classes\MortProdTypeOfBuyerTable.cls
	SUPERVISOR\Super\Classes\BuildingAndContentsTable.cls
	SUPERVISOR\Super\Classes\PayProtProductsTable.cls
	SUPERVISOR\Super\Classes\PanelValuerTable.cls
	SUPERVISOR\Super\Classes\PanelLegalRepTable.cls
	SUPERVISOR\Super\Classes\OrgUserTable.cls
	SUPERVISOR\Super\Classes\MortProdEmpEligTable.cls
	SUPERVISOR\Super\Classes\ErrorMessageTable.cls
	SUPERVISOR\Super\Classes\OtherFeeTable.cls
	SUPERVISOR\Super\Classes\MortgageLenderParametersTable.cls
	SUPERVISOR\Super\Classes\MortgageProductParametersTable.cls
	SUPERVISOR\Super\Classes\MortProdLanguageTable.cls
	SUPERVISOR\Super\Classes\MortProdPurpOfLoanTable.cls
	SUPERVISOR\Super\Classes\MortProdTypeofAppEligTable.cls
	SUPERVISOR\Super\Classes\SupervisorObjectInstanceTable.cls
	SUPERVISOR\Super\Classes\WorkingHoursTable.cls
	SUPERVISOR\Super\Classes\InvalidProductTable.cls
	SUPERVISOR\Super\Classes\OtherFeesTable.cls
	SUPERVISOR\Super\Classes\CountryTable.cls
	SUPERVISOR\Super\Classes\MortProdPropLocTable.cls
	SUPERVISOR\Super\Classes\UnitRegionTable.cls
	SUPERVISOR\Super\Classes\QualificationTable.cls
	SUPERVISOR\Super\Classes\UserCompetencyTable.cls
	SUPERVISOR\Super\Classes\UserRoleTable.cls
	SUPERVISOR\Super\Classes\ApplicationTable.cls
	SUPERVISOR\Super\Classes\PaymentProcessingTable.cls
	SUPERVISOR\Super\Classes\TemplateTable.cls
	SUPERVISOR\Super\Classes\ProductGroupingTable.cls
	SUPERVISOR\Super\Classes\MortLenderDirTable.cls
	SUPERVISOR\Super\Classes\MortProdIncentiveTable.cls
	SUPERVISOR\Super\Classes\DepartmentContactTable.cls
	SUPERVISOR\Super\Classes\UnitContactDetailsTable.cls
	SUPERVISOR\Super\Classes\BandedTable.cls
	SUPERVISOR\Super\Classes\MortProdChannelEligTable.cls

	SUPERVISOR\Super\Classes\AvailableTemplatesTable.cls
	SUPERVISOR\Super\Classes\DistributionChannelTable.cls
	SUPERVISOR\Super\Classes\AppStageTable.cls
	SUPERVISOR\Super\Classes\ApplicationLockTable.cls
	SUPERVISOR\Super\Classes\TaskLinkTable.cls
	SUPERVISOR\Super\Classes\CustomerLockTable.cls
	SUPERVISOR\Super\Classes\TaskInterfaceMessageTable.cls
	SUPERVISOR\Super\Classes\TaskInterfaceSubMessageTable.cls
	SUPERVISOR\Super\Classes\IntermediarySearch.cls
	SUPERVISOR\Super\Classes\IntermediaryProcurationFeeTable.cls
	SUPERVISOR\Super\Classes\IntermediaryOrganisation.cls
	SUPERVISOR\Super\Classes\IntermediaryBankAccountTable.cls
	SUPERVISOR\Super\Classes\IntermediaryReportDaysTable.cls
	SUPERVISOR\Super\Classes\IntermediaryCorrespondenceTable.cls
	SUPERVISOR\Super\Classes\IntermediaryCrossSellingTable.cls
	SUPERVISOR\Super\Classes\IntermediaryTargetTable.cls
	SUPERVISOR\Super\Classes\IntermediaryProductPeriodtable.cls
	SUPERVISOR\Super\Classes\IntermediaryProcFeeSplitForIntTable.cls
	SUPERVISOR\Super\Classes\IntermediaryProcFeeSplitTable.cls
	SUPERVISOR\Super\Classes\Omiga4Support.cls
	SUPERVISOR\Super\Classes\IntermediaryTable.cls
	SUPERVISOR\Super\Classes\VersionControl.cls
	SUPERVISOR\Super\Classes\ValuationFeeSetTable.cls
	SUPERVISOR\Super\Classes\BaseRateSetTable.cls
	SUPERVISOR\Super\Classes\AdminFeeSetTable.cls
	SUPERVISOR\Super\Classes\SupervisorConnections.cls
	SUPERVISOR\Super\Classes\ContactDetailsTelephoneTable.cls
	SUPERVISOR\Super\Classes\LenderOtherFees.cls
	SUPERVISOR\Super\Classes\ContactDetailsCS.cls
	SUPERVISOR\Super\Classes\LenderParametersCS.cls
	SUPERVISOR\Super\Classes\LegalFeesCS.cls
	SUPERVISOR\Super\Classes\OtherFeesCS.cls
	SUPERVISOR\Super\Classes\MIGRateSetsCS.cls
	SUPERVISOR\Super\Classes\Lender_CS.cls
	SUPERVISOR\Super\Classes\ErrorHandling.cls
	SUPERVISOR\Super\Classes\MIGRateSetsTable.cls

	SUPERVISOR\Super\Classes\LegalFeeSetTable.cls
	SUPERVISOR\Super\Classes\MortProdIncentiveCS.cls
	SUPERVISOR\Super\Classes\MortProdIntRatesCS.cls
	SUPERVISOR\Super\Classes\MortProdOtherFeeCS.cls
	SUPERVISOR\Super\Classes\MortProdParamsCS.cls
	SUPERVISOR\Super\Classes\MortProdPropLocationCS.cls
	SUPERVISOR\Super\Classes\MortProdPurpOfLoanCS.cls
	SUPERVISOR\Super\Classes\MortProdChannelEligCS.cls
	SUPERVISOR\Super\Classes\MortProdEmpEligCS.cls
	SUPERVISOR\Super\Classes\MortProdSpecialGroupCS.cls
	SUPERVISOR\Super\Classes\MortProdTypeOfBuyerCS.cls
	SUPERVISOR\Super\Classes\MortProdAdminFeesCS.cls
	SUPERVISOR\Super\Classes\MortProdValuationFeeCS.cls
	SUPERVISOR\Super\Classes\MortProdBaseRatesCS.cls
	SUPERVISOR\Super\Classes\MortProdTypicalAprCS.cls
	SUPERVISOR\Super\Classes\MortProdTypeOfAppCS.cls
	SUPERVISOR\Super\Classes\ObjectDependenciesCS.cls
	SUPERVISOR\Super\Classes\BatchAuditTable.cls
	SUPERVISOR\Super\Forms\frmEditIncomeFactors.frm
	SUPERVISOR\Super\Classes\IncomeFactorCS.cls
	SUPERVISOR\Super\Classes\IntermediaryIndividual.cls
	SUPERVISOR\Super\Classes\OrgUserContactTable.cls
	SUPERVISOR\Super\Classes\LenderPaymentDetails.cls
	SUPERVISOR\Super\Classes\LenderPaymentDetailsCS.cls
	SUPERVISOR\Super\Classes\RedemptionFeeSetTable.cls
	SUPERVISOR\Super\Classes\MPMigRateSetTable.cls
	SUPERVISOR\Super\Classes\MortProdProdCondTable.cls
	SUPERVISOR\Super\Forms\frmNewCombo.frm
	SUPERVISOR\Super\Forms\frmText.frm
	SUPERVISOR\Super\Classes\DocumentLock.cls
	SUPERVISOR\Super\Classes\MortProdAddtFeaTable.cls
	SUPERVISOR\Super\Classes\InsuranceAdminFeeSetTable.cls
	SUPERVISOR\Super\Classes\ProductSwitchFeeSetTable.cls
	SUPERVISOR\Super\Classes\TTFeeSetTable.cls
	SUPERVISOR\Super\Classes\MortProdTTFeesCS.cls
	SUPERVISOR\Super\Classes\MortProdIAFeesCS.cls

	SUPERVISOR\Super\Classes\MortProdSwitchFeesCS.cls
	SUPERVISOR\Super\Classes\RentalIncomeRateSetTable.cls
	SUPERVISOR\Super\Classes\PackControlTable.cls
	SUPERVISOR\Super\Classes\PackMemberTable.cls
	SUPERVISOR\Super\Classes\LinkedTaskTable.cls
	SUPERVISOR\Super\Classes\AdditionalBorrowingFeeSetTable.cls
	SUPERVISOR\Super\Classes\CreditLimitIncreaseFeeSetTable.cls
	SUPERVISOR\Super\Classes\MortProdAddBorrFeeCS.cls
	SUPERVISOR\Super\Classes\MortProdCredLimIncFeeCS.cls
	SUPERVISOR\Super\Classes\FirmPermissionsTable.cls
	SUPERVISOR\Super\Classes\FirmTradingNameTable.cls
	SUPERVISOR\Super\Classes\FirmActivityTable.cls
	SUPERVISOR\Super\Classes\FirmBrokerTable.cls
	SUPERVISOR\Super\Classes\IndividualPackagerTable.cls
	SUPERVISOR\Super\Classes\ActivityFSATable.cls
	SUPERVISOR\Super\Classes\MortgageProductExclusivityTable.cls
	SUPERVISOR\Super\Classes\MortProdProdClassTable.cls
	SUPERVISOR\Super\Classes\MortProdNatureOfLoanTable.cls
	SUPERVISOR\Super\Classes\MortProdIncomeStatusTable.cls
	SUPERVISOR\Super\Classes\IntroducerTable.cls
	SUPERVISOR\Super\Classes\TransferOfEquityFeeSetTable.cls
	SUPERVISOR\Super\Classes\MortProdTranEquFeeCS.cls
	SUPERVISOR\Super\Classes\TaskTable.cls
	SUPERVISOR\Super\Classes\ActivityTable.cls
	SUPERVISOR\Super\Classes\ARBrokerTable.cls
	SUPERVISOR\Super\Classes\ARFirmTable.cls
	SUPERVISOR\Super\Classes\AssociationTable.cls
	SUPERVISOR\Super\Classes\ClubTable.cls
	SUPERVISOR\Super\Classes\DABrokerTable.cls
	SUPERVISOR\Super\Classes\FirmPackagerTable.cls
	SUPERVISOR\Super\Classes\MortgageClubNetworkAssociationTable.cls
	SUPERVISOR\Super\Classes\IntroducerFirmTable.cls
	SUPERVISOR\Super\Classes\PrincipalFirmTable.cls
	SUPERVISOR\Super\Forms\frmBatchErrors.frm
	SUPERVISOR\Super\Forms\frmLenderDetails.frm
	SUPERVISOR\Super\Classes\FirmClubNetworkAssociationTable.cls

	SUPERVISOR\Super\Classes\AppointmentsTable.cls
	SUPERVISOR\Super\Classes\SQLAssistSP.cls
	SUPERVISOR\Super\Classes\SupervisorUpdateTable.cls
	SUPERVISOR\Super\Forms\frmAddPackMember.frm
	SUPERVISOR\Super\Forms\frmFindDepartments.frm
	SUPERVISOR\Super\Forms\frmEditGlobal.frm
	SUPERVISOR\Super\Forms\frmEditRegion.frm
	SUPERVISOR\Super\Forms\frmEditWorkingHoursType.frm
	SUPERVISOR\Super\Forms\frmErrors.frm
	SUPERVISOR\Super\Forms\frmEditCurrencies.frm
	SUPERVISOR\Super\Forms\frmEditTargetDetails.frm
	SUPERVISOR\Super\Forms\frmAddEditProductPeriod.frm
	SUPERVISOR\Super\Forms\frmEditBuildingsAndContents.frm
	SUPERVISOR\Super\Forms\frmPromotionUser.frm
	SUPERVISOR\Super\Classes\DisbursementPaymentTable.cls
	SUPERVISOR\Super\Classes\AppFactFindTable.cls
	SUPERVISOR\Super\Classes\NonWorkingDayTable.cls
	SUPERVISOR\Super\Classes\PanelBankAccountTable.cls
	SUPERVISOR\Super\Classes\PanelValuerTypeTable.cls
	SUPERVISOR\Super\Classes\UserQualification.cls
	SUPERVISOR\Super\Classes\BandedGlobalParametersTable.cls
	SUPERVISOR\Super\Classes\WorkingHourTypeTable.cls
	SUPERVISOR\Super\Classes\MortProdSpecialGroupTable.cls
	SUPERVISOR\Super\Classes\MortProdTypicalAPRTable.cls
	SUPERVISOR\Super\Classes\AdditionalLenderParameters.cls
	SUPERVISOR\Super\Classes\BankHolidayTable.cls
	SUPERVISOR\Super\Classes\ActionOwnerTable.cls
	SUPERVISOR\Super\Classes\ComboValueGroupTable.cls
	SUPERVISOR\Super\Classes\RateTable.cls
	SUPERVISOR\Super\Classes\TaskPriority.cls
	SUPERVISOR\Super\Classes\ComboValidationTable.cls
	SUPERVISOR\Super\Classes\MortProdIntRatesTable.cls
	SUPERVISOR\Super\Classes\PanelBankAccountDetails.cls
	SUPERVISOR\Super\Classes\DepartmentTable.cls
	SUPERVISOR\Super\Classes\BusinessGroupTable.cls
	SUPERVISOR\Super\Classes\LedgerCodesTable.cls

	SUPERVISOR\Super\Classes\RegionTable.cls
	SUPERVISOR\Super\Classes\ComboUtils.cls
	SUPERVISOR\Super\Classes\StageTable.cls
	SUPERVISOR\Super\Classes\BatchJobsTable.cls
	SUPERVISOR\Super\Classes\ComboValueTable.cls
	SUPERVISOR\Super\Classes\BatchProcessing.cls
	SUPERVISOR\Super\Classes\LenderDetailsCS.cls
	SUPERVISOR\Super\Classes\AddressTable.cls
	SUPERVISOR\Super\Classes\ContactDetailsTable.cls
	SUPERVISOR\Super\Classes\LegalFeesTable.cls
	SUPERVISOR\Super\Classes\MIGRateTable.cls
	SUPERVISOR\Super\Classes\LifeCoverRatesTable.cls
	SUPERVISOR\Super\Classes\AdditionalMortgageProductParams.cls
	SUPERVISOR\Super\Classes\MortProdSpecialGroup.cls
	SUPERVISOR\Super\Classes\MortProdTypicalAPR.cls
	SUPERVISOR\Super\Classes\PromoteCS.cls
	SUPERVISOR\Super\Classes\AllowableIncomeFactorTable.cls
	SUPERVISOR\Super\Classes\CompetencyTable.cls
	SUPERVISOR\Super\Classes\IncomeMultipleSetTable.cls
	SUPERVISOR\Super\Classes\CompetencyAuditTable.cls
	SUPERVISOR\Super\Forms\frmAddComboGroup.frm
	SUPERVISOR\Super\Forms\frmUserRole.frm
	SUPERVISOR\Super\Classes\ComboValidationAuditTable.cls
	SUPERVISOR\Super\Classes\ComboValueAuditTable.cls
	SUPERVISOR\Super\Forms\frmEditPrintingDocument.frm
	SUPERVISOR\Super\Classes\IntroducerSearch.cls
	SUPERVISOR\Super\Forms\frmEditBrokerTradingName.frm
	SUPERVISOR\Super\Forms\frmEditPackagerTradingName.frm
	SUPERVISOR\Super\Classes\FirmBrokerSearch.cls
	SUPERVISOR\Super\Classes\IntroducerDAIndividualTable.cls
	SUPERVISOR\Super\Classes\IntroducerPackagerIndividualTable.cls
	SUPERVISOR\Super\Classes\ProcFeeDefaultTable.cls
	SUPERVISOR\Super\Forms\frmEditDefaultProcurationFees.frm
	SUPERVISOR\Super\Forms\frmEditLoanAmountAdjustments.frm
	SUPERVISOR\Super\Forms\frmEditLTVAmountAdjustments.frm
	SUPERVISOR\Super\Classes\ProcFeeAdjByLoanTable.cls

	SUPERVISOR\Super\Classes\ProcFeeAdjByLTVTable.cls
	SUPERVISOR\Super\Classes\ConditionsTable.cls
	SUPERVISOR\Super\Classes\ARFirmSearch.cls
	SUPERVISOR\Super\Classes\AssociationSearch.cls
	SUPERVISOR\Super\Classes\ClubSearch.cls
	SUPERVISOR\Super\Classes\DABrokerSearch.cls
	SUPERVISOR\Super\Classes\FirmPackagerSearch.cls
	SUPERVISOR\Super\Classes\IndividualPackagerSearch.cls
	SUPERVISOR\Super\Classes\IntroducerPackagerTable.cls
	SUPERVISOR\Super\Classes\PrincipalFirmSearch.cls
	SUPERVISOR\Super\Classes\ARBrokerSearch.cls
	SUPERVISOR\Super\Forms\frmMaintainFirmLinks .frm
	SUPERVISOR\Super\Classes\IntroducerARFirmTable.cls
	SUPERVISOR\Super\Classes\IntroducerDAFirmTable.cls
	SUPERVISOR\Super\Classes\IntroducerARIndividualTable.cls
	SUPERVISOR\Super\Forms\frmProductErrors.frm
	SUPERVISOR\Super\Classes\PrintingTemplateTable.cls
	SUPERVISOR\Super\Classes\TableAccess.cls
	SUPERVISOR\Super\Forms\frmProductDetails.frm
	SUPERVISOR\Super\Classes\UserRole.cls
	SUPERVISOR\Super\Forms\frmMaintainFSAAssociations.frm
	SUPERVISOR\Super\Classes\dataaccess.cls
	SUPERVISOR\Super\Forms\frmEditPaymentForCompletions.frm
	SUPERVISOR\Super\Forms\frmEditGlobalBanded.frm
	SUPERVISOR\Super\Forms\frmEditIncentives.frm
	SUPERVISOR\Super\Forms\frmFindApplication.frm
	SUPERVISOR\Super\Forms\frmEditPaymentProcessing.frm
	SUPERVISOR\Super\Forms\frmEditRate.frm
	SUPERVISOR\Super\Forms\frmEditQuestions.frm
	SUPERVISOR\Super\Forms\frmEditCountry.frm
	SUPERVISOR\Super\Forms\frmEditCombo.frm
	SUPERVISOR\Super\Forms\frmEditMIGRates.frm
	SUPERVISOR\Super\Forms\frmLegalFeeSets.frm
	SUPERVISOR\Super\Forms\frmEditLifeCover.frm
	SUPERVISOR\Super\Forms\frmFindPanel.frm
	SUPERVISOR\Super\Classes\GlobalParameter.cls

	SUPERVISOR\Super\Classes\UserCompetency.cls
	SUPERVISOR\Super\Classes\ApplicationProcessing.cls
	SUPERVISOR\Super\Classes\DatabaseOptions.cls
	SUPERVISOR\Super\Classes\LegalRepDetails.cls
	SUPERVISOR\Super\Classes\ValuerDetails.cls
	SUPERVISOR\Super\Classes\MortProdIncIncentiveTable.cls
	SUPERVISOR\Super\Classes\SupervisorConnectionTable.cls
	SUPERVISOR\Super\Classes\SupervisorObjectTable.cls
	SUPERVISOR\Super\Classes\StageTaskTable.cls
	SUPERVISOR\Super\Classes\CurrencyTable.cls
	SUPERVISOR\Super\Classes\IntermediaryBankDetails.cls
	SUPERVISOR\Super\Classes\AdminFeeTable.cls
	SUPERVISOR\Super\Classes\BaseRateTable.cls
	SUPERVISOR\Super\Classes\PayProtRatesTable.cls
	SUPERVISOR\Super\Classes\MortProdValuationFee.cls
	SUPERVISOR\Super\Classes\ThirdPartyDetails.cls
	SUPERVISOR\Super\Classes\NameAndAddressDirTable.cls
	SUPERVISOR\Super\Classes\UnitTable.cls
	SUPERVISOR\Super\Forms\frmEditCompetency.frm
	SUPERVISOR\Super\Forms\frmEditConditions.frm
	SUPERVISOR\Super\Classes\RedemptionFeeTable.cls
	SUPERVISOR\Super\Classes\MPMigRateTable.cls
	SUPERVISOR\Super\Classes\MortProdProdCond.cls
	SUPERVISOR\Super\Forms\frmEditIncMultiples.frm
	SUPERVISOR\Super\Forms\frmEditRedemFeeSets.frm
	SUPERVISOR\Super\Forms\frmFindUsers.frm
	SUPERVISOR\Super\Classes\InsuranceAdminFeeBand.cls
	SUPERVISOR\Super\Classes\ProductSwitchFeeTable.cls
	SUPERVISOR\Super\Classes\TTFeeBand.cls
	SUPERVISOR\Super\Classes\RentalIncomeRateSetBandTable.cls
	SUPERVISOR\Super\Classes\AdditionalBorrowingFeeTable.cls
	SUPERVISOR\Super\Classes\CreditLimitIncreaseFeeTable.cls
	SUPERVISOR\Super\Classes\MortProdAddBorrFee.cls
	SUPERVISOR\Super\Classes\MortProdCredLimIncFee.cls
	SUPERVISOR\Super\Classes\ValuationFeeTable.cls
	SUPERVISOR\Super\Classes\OmigaUnitTable.cls

	SUPERVISOR\Super\Classes\IntroUnitTable.cls
	SUPERVISOR\Super\Classes\TransferOfEquityFeeTable.cls
	SUPERVISOR\Super\Classes\MortProdTranEquFee.cls
	SUPERVISOR\Super\Modules\global.bas
	SUPERVISOR\Super\Classes\OmigaUserTable.cls
	SUPERVISOR\Super\Classes\AdditionalQuestionsTable.cls
	SUPERVISOR\Super\Classes\PanelTable.cls
	SUPERVISOR\Super\Forms\frmEditAssociation.frm
	SUPERVISOR\Super\Forms\frmEditIntCrossSelling.frm
	SUPERVISOR\Super\Forms\frmFindConditions.frm
	SUPERVISOR\Super\Classes\MortProdProdClass.cls
	SUPERVISOR\Super\Classes\MortProdNatureOfLoan.cls
	SUPERVISOR\Super\Classes\MortProdIncomeStatus.cls
	SUPERVISOR\Super\Forms\frmEditPayProtProducts.frm
	SUPERVISOR\Super\Forms\frmMainThirdParty.frm
	SUPERVISOR\Super\Forms\frmFindLocks.frm
	SUPERVISOR\Super\Forms\frmEditBatch.frm
	SUPERVISOR\Super\Forms\frmEditActivities.frm
	SUPERVISOR\Super\Forms\frmFindProducts.frm
	SUPERVISOR\Super\Classes\TaskOwner.cls
	SUPERVISOR\Super\Classes\ContactDetails.cls
	SUPERVISOR\Super\Classes\LegalFees.cls
	SUPERVISOR\Super\Classes\MIGRateSets.cls
	SUPERVISOR\Super\Classes\MortProdPurpOfLoan.cls
	SUPERVISOR\Super\Classes\MortProdTypeofAppElig.cls
	SUPERVISOR\Super\Classes\MortProdChannelElig.cls
	SUPERVISOR\Super\Classes\MortProdEmpElig.cls
	SUPERVISOR\Super\Classes\MortProdAdminFees.cls
	SUPERVISOR\Super\Classes\MortProdOtherFee.cls
	SUPERVISOR\Super\Classes\MortProdBaseRates.cls
	SUPERVISOR\Super\Classes\MortProdPropLocation.cls
	SUPERVISOR\Super\Classes\EditInterestRateCS.cls
	SUPERVISOR\Super\Classes\IncomeFactor.cls
	SUPERVISOR\Super\Forms\frmEditMPMigRates.frm
	SUPERVISOR\Super\Forms\frmEditRentalIncomeRate.frm
	SUPERVISOR\Super\Forms\frmAdditionalFeatures.frm

	SUPERVISOR\Super\Classes\MortProdTTFees.cls
	SUPERVISOR\Super\Classes\MortProdIAFees.cls
	SUPERVISOR\Super\Classes\MortProdSwitchFees.cls
	SUPERVISOR\Super\Classes\LenderDetails.cls
	SUPERVISOR\Super\Classes\MortgageLendersTable.cls
	SUPERVISOR\Super\Classes\MortProdTypeOfBuyer.cls
	SUPERVISOR\Super\Forms\frmEditIntTargetDetails.frm
	SUPERVISOR\Super\Forms\frmEditStages.frm
	SUPERVISOR\Super\Forms\frmFindFirmBroker.frm
	SUPERVISOR\Super\Forms\frmFindIntermediary.frm
	SUPERVISOR\Super\Classes\MortgageProductDetails.cls
	SUPERVISOR\Super\Forms\frmEditBroker.frm
	SUPERVISOR\Super\Classes\MortgageProductCS.cls
	SUPERVISOR\Super\Forms\frmEditPrintingPack.frm
	SUPERVISOR\Super\Forms\frmeditadminfees.frm
	SUPERVISOR\Super\Forms\frmEditBaseRates.frm
	SUPERVISOR\Super\Forms\frmEditValuation.frm
	SUPERVISOR\Super\Classes\PasswordTable.cls
	SUPERVISOR\Super\Classes\BatchTable.cls
	SUPERVISOR\Super\Classes\MortProdIncentive.cls
	SUPERVISOR\Super\Classes\EditInterestRate.cls
	SUPERVISOR\Super\Classes\MortProdIntRates.cls
	SUPERVISOR\Super\Forms\frmEditProductSwitchFee.frm
	SUPERVISOR\Super\Forms\frmEditInsuranceAdminFee.frm
	SUPERVISOR\Super\Forms\frmEditTTFee.frm
	SUPERVISOR\Super\Forms\frmEditCreditLimitIncreaseFees.frm
	SUPERVISOR\Super\Forms\frmEditTransferOfEquityFees.frm
	SUPERVISOR\Super\Forms\frmEditAdditionalBorrowingFees.frm
	SUPERVISOR\Super\Forms\frmDatabaseOptions.frm
	SUPERVISOR\Super\Forms\frmEditBatchInfo.frm
	SUPERVISOR\Super\Forms\frmEditIntReportDetails.frm
	SUPERVISOR\Super\Forms\frmFindARBroker.frm
	SUPERVISOR\Super\Forms\frmFindARFirm.frm
	SUPERVISOR\Super\Forms\frmFindAssociation.frm
	SUPERVISOR\Super\Forms\frmFindClubs.frm
	SUPERVISOR\Super\Forms\frmFindDABroker.frm

	SUPERVISOR\Super\Forms\frmFindIndividualPackager.frm
	SUPERVISOR\Super\Forms\frmFindPrincipalFirm.frm
	SUPERVISOR\Super\Forms\frmFindFirmPackager.frm
	SUPERVISOR\Super\Forms\frmEditPrintingTemplate.frm
	SUPERVISOR\Super\Forms\frmEditStageTask.frm
	SUPERVISOR\Super\Forms\frmMaintainProductExclusivity.frm
	SUPERVISOR\Super\Forms\frmEditBusinessGroup.frm
	SUPERVISOR\Super\Forms\frmEditDistChannel.frm
	SUPERVISOR\Super\Forms\frmEditProdProcurationFee.frm
	SUPERVISOR\Super\Classes\LockProcessing.cls
	SUPERVISOR\Super\Forms\frmLogin.frm
	SUPERVISOR\Super\Classes\IntermediaryScreenDetials.cls
	SUPERVISOR\Super\Forms\frmEditProcurationFees.frm
	SUPERVISOR\Super\Classes\MortgageProductTable.cls
	SUPERVISOR\Super\Forms\frmEditDepartment.frm
	SUPERVISOR\Super\Forms\frmEditPayProtRates.frm
	SUPERVISOR\Super\Classes\Intermediary.cls
	SUPERVISOR\Super\Classes\MortgageProductDetailsCS.cls
	SUPERVISOR\Super\Forms\frmEditThirdParty.frm
	SUPERVISOR\Super\Classes\FormProcessing.cls
	SUPERVISOR\Super\Classes\IntermediaryProcFeeTabHandler.cls
	SUPERVISOR\Super\Forms\frmMain.frm
	SUPERVISOR\Super\Classes\Lender.cls
	SUPERVISOR\Super\Forms\frmEditUser.frm
	SUPERVISOR\Super\Forms\frmEditPackager.frm
	SUPERVISOR\Super\Classes\HandleUpdates.cls
	SUPERVISOR\Super\Classes\OmigaUser.cls
	SUPERVISOR\Super\Forms\frmEditIntermediaries.frm
	SUPERVISOR\Super\Forms\frmEditUnit.frm
	SUPERVISOR\Super\Forms\frmEditTasks.frm
	SUPERVISOR\Super\Forms\frmEditIndividualARBroker.frm
	SUPERVISOR\Super\Forms\frmEditIndividualDABroker.frm
	SUPERVISOR\Super\Forms\frmEditIndividualPackager.frm
	SUPERVISOR\Super\Classes\MortgageProduct.cls
	SUPERVISOR\Super\Forms\frmCopyProducts.frm
	SUPERVISOR\Super\Forms\frmManageUpdates.frm

	SUPERVISOR\Super\Classes\SupervisorObjectCopy.cls
	SUPERVISOR\Super\Classes\MainSuper.cls

13 Appendix D – Unsupported Methods

The following table contains the methods and properties of the COM components that are not supported by the recommended target components:

MSDataGridLib.DataGrid
MSDataGridLib.Column.Left
MSDataGridLib.Column.Width
MSDataGridLib.DataGrid.AddNewMode
MSDataGridLib.DataGrid.Align
MSDataGridLib.DataGrid.AllowUpdate
MSDataGridLib.DataGrid.DataBindings
MSDataGridLib.DataGrid.DataFormats
MSDataGridLib.DataGrid.DefColWidth
MSDataGridLib.DataGrid.DragIcon
MSDataGridLib.DataGrid.DragMode
MSDataGridLib.DataGrid.ErrorText
MSDataGridLib.DataGrid.HeadLines
MSDataGridLib.DataGrid.HelpContextID
MSDataGridLib.DataGrid.MarqueeStyle
MSDataGridLib.DataGrid.Object
MSDataGridLib.DataGrid.RecordSelectors
MSDataGridLib.DataGrid.RowDividerStyle
MSDataGridLib.DataGrid.RowTop
MSDataGridLib.DataGrid.Scroll
MSDataGridLib.DataGrid.SelLength
MSDataGridLib.DataGrid.SelStart
MSDataGridLib.DataGrid.SelText
MSDataGridLib.DataGrid.Splits
MSDataGridLib.DataGrid.TabAcrossSplits
MSDataGridLib.DataGrid.TabAction
MSDataGridLib.DataGrid.WrapCellPointer
MSDataGridLib.SelBookmarks.Add
MSDataGridLib.SelBookmarks.Count
MSDataGridLib.SelBookmarks.Remove

MSDataGridLib.DataGrid.AllowArrows
MSDataGridLib.Splits.SizeMode
MSDataListLib.DataCombo
MSDataListLib.DataCombo.Appearance
MSDataListLib.DataCombo.BoundText
MSDataListLib.DataCombo.DataBindings
MSDataListLib.DataCombo.DataField
MSDataListLib.DataCombo.DataFormat
MSDataListLib.DataCombo.DataMember
MSDataListLib.DataCombo.DragIcon
MSDataListLib.DataCombo.DragMode
MSDataListLib.DataCombo.HelpContextID
MSDataListLib.DataCombo.ListField
MSDataListLib.DataCombo.MouseIcon
MSDataListLib.DataCombo.OLEDragMode
MSDataListLib.DataCombo.OLEDropMode
MSDataListLib.DataCombo.RowMember
MSDataListLib.DataCombo.RowSource
MSDataListLib.DataCombo.Style
MSDataListLib.DataCombo.WhatsThisHelpID