




Technical Analysis & Migration Proposal

Omiga
Vertex
 The Vertex logo consists of a blue stylized 'V' with a horizontal line through its center, and the word "VERTEX" in a bold, blue, sans-serif font below it.

Presented by ArtinSoft Consulting Services

Version 2.4

5/29/2007

Legal Notice

The information contained in this document is a representation of ArtinSoft's view on the issues to be discussed as of the date of publication and is subject to change at any time without notice. This document and its contents are provided by AIS without warranty of any kind, and should not be interpreted as an offer or commitment on the part of ArtinSoft.

All trademarks are the property of their respective owners.

©2007 ArtinSoft. All rights reserved.

Contents

1	Introduction	5
2	Executive Summary	5
3	Restrictions and Assumptions	7
4	Glossary of Terms.....	8
5	Technical Analysis.....	9
5.1	Methodology.....	9
5.2	Application Background.....	9
5.3	Application Structure, Operation and Deployment	10
5.4	Support, Development and Maintenance.....	12
5.5	Testing Processes and Structure	12
5.6	Design Assessment.....	12
5.6.1	Current Environment.....	12
5.7	Code Assessment	13
5.7.1	Modularity	13
5.7.2	Size	14
5.7.3	Complexity	16
5.8	Migration Strategy.....	16
5.9	Migration Requirements	17
5.9.1	Migration of Data Access Technology from ADO to ADO.NET	17
5.9.2	Replacement of other COM controls and components, when possible.....	17
5.9.3	MSXML to System.Xml.....	17
5.9.4	Change the VB6 namespace functions to native .NET functions.	18
5.9.5	Use of Shared classes instead of Modules	18
5.9.6	Error handling conversion modifications	18
6	Migration Proposal	22
6.1.1	Proposed Target Environment.....	22
6.1.2	Target Architecture Description.....	22
6.1.3	Use of Visual Basic Upgrade Companion	22
	Add COM Support.....	23
	COM+ and Transactions	23
1.	Adding COM+ Support.....	24
	Interfaces.....	25
	AssemblyInfo.cs	25
	Porting Main Method	26
	Porting the App Object	26
	Rewrite Late Binding to COM Classes.....	27
	Dispose and Using	28
6.1.4	Items to be addressed from Visual Basic 6.0.....	30
6.2	Project Structure & Responsibilities.....	32
6.2.1	Visual Basic Upgrade Companion Customization (VBUCC)	34
6.2.2	VB6 Changes and QA	35

6.2.3	Migration	35
6.2.4	Quality Assurance (QA)	37
6.2.5	Ad-hoc Quality Assurance (Ad-hoc QA)	37
6.2.6	Bug Fixing	37
6.2.7	Performance Testing (Perf. Testing).....	37
6.2.8	Performance Tuning (Perf. Tuning)	37
6.2.9	C# Enhancements (C#)	37
6.3	Business Case.....	38
6.4	Final Recommendation	38
6.5	Additional recommendations	40
6.6	Time Estimation	41
6.6.1	Visual Basic Upgrade Companion Customization	41
6.6.2	Migration	41
6.7	Anticipated Project Budget.....	42
6.7.1	Budget for Visual Basic Upgrade Companion License	42
6.7.2	Budget for Visual Basic Upgrade Companion Customization.....	42
6.7.3	Budget for Migration	43
6.7.4	Overall Migration Budget.....	43
6.8	Appendix A: Client Statistics	44
6.9	Appendix B: Core Statistics.....	48

1 Introduction

This document intends to provide an analysis of Omiga, Vertex's mortgage management application. The application is mainly written in Visual Basic 6.0, and is used by customers of Vertex in the United Kingdom.

ArtinSoft performed a 22-days analysis, with two main objectives:

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

This analysis took place at the Vertex headquarters located in Cheltenham, England and at ArtinSoft's offices in San José, Costa Rica. A certain number of ArtinSoft consultants as well as members of the Vertex development and testing teams were involved in the process.

This is a confidential document, and it is intended for review by Vertex and third party consultants that Vertex may consider necessary.

2 Executive Summary

Vertex has several mortgage applications developed in Visual Basic 6.0 and is currently developing new applications and web sites in the .NET environment. In the case of Omiga, the application is structured as a well-defined layered application based upon an ASP user interface with a database back-end and some server components that contain the business logic.

Vertex plans to migrate these applications to C# .NET, maintaining the current architecture of the application, but replacing coding patterns and best practices to adjust the Visual Basic Code to C#.

One of the most important changes expected to be accomplished during the migration process is the definition of a common component that includes all the shared code in the VB6 projects to unify and avoid redundant code.

The course of action recommended by ArtinSoft is to migrate the 77 VB6 projects to C# .NET using a customized version of the ArtinSoft Visual Basic Upgrade Companion. The customization of this migration tool will enable Vertex to automatically convert coding patterns and meet its preferred standards, as defined on the "Using VB62CS" document previously provided by Vertex.

The automatic migration and developer testing stages will be performed by ArtinSoft resources. Vertex expects to have a non-tested version in about 3 months to introduce new customer requirements and updates to the code

generated in C#. This change will be made on a copy of the code, while the original codebase will continue to be tested and fixed as in any regular migration project.

The projects will be separated into two different groups; Client and Core. The common component that Vertex intends to create contains all the shared modules and classes, and the output of this migration will be one or more libraries to be used by both Client and Core.

The Core group has the majority of shared files, and the Client has an extension to those files that could represent new functionality or business rules required for a specific customer. The objective is to consolidate a single version by merging both Client and Core files.

This document presents an estimate for two different project execution models to be chosen by Vertex:

- **Migration Plan 1:** In this scenario, development of new features in VB6 from Vertex will stop until the migration project finishes, using ArtinSoft regular migration strategies. This scenario will have a duration of 8 calendar-months for Migration and Developer Testing, and 2 calendar-months for Testing, resulting in a total of 10 calendar-months for full Functional Equivalence.
- **Migration Plan 2:** This scenario adjusts the time to three months for the Automatic Migration and Developer Testing phases and afterwards, creating a copy of the Green Code to continue working on adding new functionality. The other copy of the code will undergo the regular Testing and Bug Fixing phases and once Functional Equivalence is achieved, both versions will be merged. This merging task will be assumed by Vertex, while ArtinSoft will be responsible for the achievement of Functional Equivalence. This scenario will have a duration of 3 calendar-months for Migration and Developer Testing, and 2 calendar-months for Testing, resulting in a total of 5 calendar-months for full Functional Equivalence.

Please refer to section "6.6 Time Estimation" 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 \$89,999.00
- Customizations to the Visual Basic Upgrade Companion to automatically convert coding patterns at a cost of \$ 27,798.00
- Migration to Functional Equivalence at a cost of \$481,003 for Migration Plan 1, or at a cost of \$582,103 for Migration Plan 2.

The overall migration budget, including VBUC license and customization, as well as the migration to Functional equivalence is \$598,800 for Migration Plan 1 and \$699,900 for Migration Plan 2.

3 Restrictions and Assumptions

1. Effort and cost estimates provided in this document are based on the Visual Basic 6.0 code delivered to ArtinSoft during the week of September 23rd, 2007. Due to expected source code changes, an adjustment to these estimates will be necessary prior to the beginning of the migration project.
2. For the duration estimate provided in section 6.6.2, the following has been assumed:
 - 2.1. All resources involved in the Manual Migration phase of the project have a reasonable level of knowledge and experience in C# .NET.
 - 2.2. All resources involved in the Manual Migration phase of the project, except resources provided by ArtinSoft, have a reasonable level of knowledge in the applications to be migrated.
3. This Ready document analyzes an eventual migration project to Functional Equivalence and Performance Testing and Tuning. No re-architecture is included in the proposed project.
4. Estimated effort for Performance Testing and Tuning is based on prior experience on other projects and may not reflect the actual effort that will be required to test and tune the applications. These estimates aim to be a recommendation of the amount of time to be invested in Performance Testing and Tuning in a worst-case scenario. Actual effort spent on these tasks may be less.

4 Glossary of Terms

API: Application Programming Interface.

COM Interop: Technology that allows a .NET application to interact with a legacy COM object or library. While this technology enables the use of legacy classes, it typically results in degraded performance.

Effective Lines of Code: Lines of code without comments and blank lines. This includes visual lines.

Feature Parity: An application that implements the same business processes as another application, but with behavioral differences. Usually one application is the source for the other.

Functional Equivalence: An application that implements the same business processes as another application while retaining the same behavior to the maximum extent possible. Usually one application is the source for the other.

Green Code: C# code as it was produced by the Visual Basic Upgrade Companion, with no manual changes applied to it.

LOC: Lines of Code, including comments and blank lines.

Mixed Team: Project Team involving resources from ArtinSoft and the Client.

Project Team: Group of Developers, Testers and Managing Personnel that makes charge of the execution of a migration project.

UI: User Interface.

VB: Microsoft Visual Basic (either version 6.0 or .NET)

C#: Microsoft C# .NET.

VB6: Microsoft Visual Basic 6.0.

VBUC: Visual Basic Upgrade Companion.

Visual Basic Upgrade Companion: ArtinSoft product that allows extension of the Visual Basic Upgrade Wizard to increase the overall migration coverage for a particular source code base.

Visual Basic Upgrade Companion Customization: Process in which ArtinSoft extends the Visual Basic Upgrade Companion to support additional libraries or components.

5 Technical Analysis

5.1 Methodology

The analysis methodology is divided into three phases:

- 1. Design assessment:** this phase includes the evaluation of current application design, to determine the advantages and disadvantages it presents for the migration process. The following aspects were taken into account:
 - 1.1.** N-Tier architecture, if any
 - 1.2.** Server-side business logic
 - 1.3.** Data repositories.
 - 1.4.** Web application, if any
- 2. Code assessment:** this phase is comprised of the evaluation of code size (in terms of lines of code) and the complexity regarding a migration process. The following aspects were taken into account:
 - 2.1.** Code size
 - 2.2.** Code complexity (for a migration process)
 - 2.3.** Code type
 - 2.4.** Presence of unsupported features
- 3. 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, whenever possible.

5.2 Application Background

Omiga is part of an important system in Vertex. It supports day-to-day mortgage operations for several costumers.

Omiga consists of two logical components: the Omiga Application and Supervisor, a tool that allows administrators to maintain configuration data. The Omiga Application communicates with a number of other applications, which are not within the scope of the current migration project.

Communication between the Omega Application and these other applications will be tested using dummy classes or stubs.

It is important to remark that performance is a key point in this application. Given that an important part of the code is related to XML operations, this area will receive special emphasis during the migration process. Vertex also aims to change legacy MSXML libraries to the native System.Xml namespace.

5.3 Application Structure, Operation and Deployment

These applications are structured as a classic ASP 3-tier system; the Components are running on the same Application Server as the Web pages, with the database running on a machine of its own.

In theory, the application should support databases different from SQL Server, but that environment will not be considered during the migration nor will it be included in the test cases scenarios supplied by Vertex.

Since the application is implemented in classic ASP, ArtinSoft will only be responsible, in terms of Functional Equivalence, for the COM+ components and libraries.

The GUI (Classic ASP) will be ported by Vertex and the components generated by the migration could interface with Classic ASP GUI.

Omega is deployed on a server and used as an intranet application. Figure 1 shows a simplified view of the current architecture.

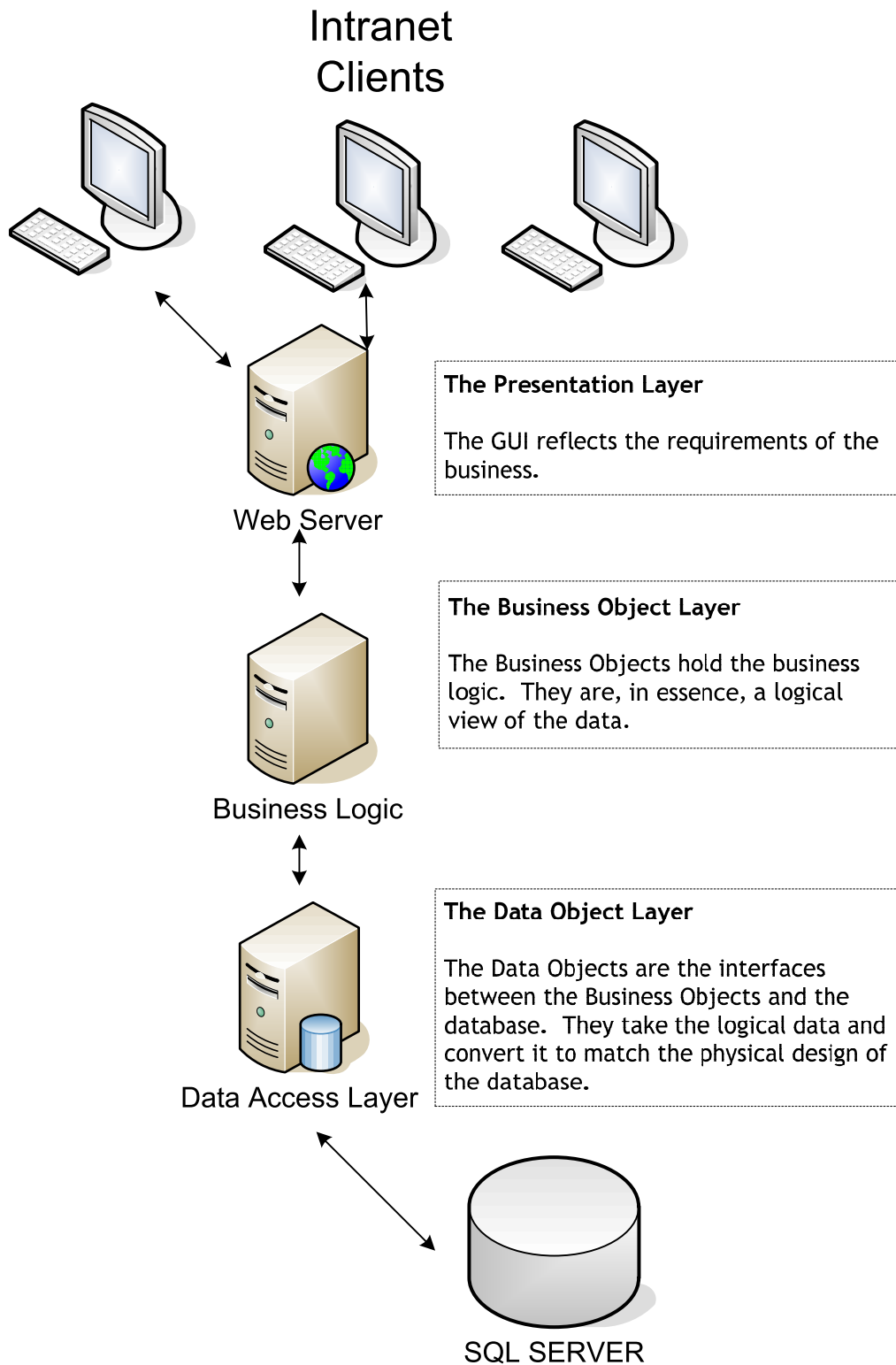


Figure 1. **Current application architecture**

5.4 Support, Development and Maintenance

The support, development and maintenance tasks are all performed by Vertex at their offices.

Currently a team of 30 developers, 10 business analysts, and 5 testers is responsible for the maintenance and enhancement of the VB6 application. The developers have been working in other .NET projects using C#; this is why Vertex has chosen to migrate to C# instead of VB.NET.

5.5 Testing Processes and Structure

Testing and quality of the applications are top-priority issues for Vertex, given the nature of the business.

As a result, Vertex has a Quality Assurance (QA) department dedicated entirely to Omiga. The staff works with manual and automatic testing using Rational Robot and practices a very strong testing procedure that includes different test cases depending on the customer.

The testing in this application is quite complicated due to its nature. Testers are very familiar with the business and this is why the test cases are not detailed as required by ArtinSoft.

During the automatic migration, Vertex will be working on the generation of new detailed test cases based on the existent documents to set up the testing environment prior to the end of the Automatic Migration phase.

Another important consideration is that Vertex will be entitled to report any bug that is out of the scope of the test cases, as long as Vertex provides the detailed information of the error. This Testing and Bug Fixing stage will have a fixed time defined by Vertex.

Given that the application architecture will not change in this project and that the main target is to achieve Functional Equivalence, the Vertex QA team does not perceive an important change in the way the .NET applications are going to be tested. In fact, Vertex will maintain the same structure, and due to the platform change, some Performance Testing will be executed.

5.6 Design Assessment

5.6.1 Current Environment

Application Server: Windows 2000 / XP

Database: Microsoft SQL Server

Database Version: 2000

5.7 Code Assessment

5.7.1 Modularity

The scope of the migration consists of the following 77 Visual Basic 6.0 projects:

axword.vbp	omRiskMatrix.vbp
CMRules.vbp	RiskMatrixRules.vbp
Delivery.vbp	omRA.vbp
eKFI.vbp	omRARules.vbp
eKFIEngine.vbp	omSub.vbp
KFIPrintProcessor.vbp	omCDRules.vbp
MsgTm.vbp	omValuationRules.vbp
ODITransformer.vbp	omWSInterface.vbp
om4To3.vbp	omDPS.vbp
omAdmin.vbp	omEXP.vbp
omAdminRules.vbp	omFVS.vbp
omAIP.vbp	omGBAddress.vbp
omApp.vbp	omGBNatRegister.vbp
omAppProc.vbp	omHI.vbp
omAPRules.vbp	omIC.vbp
omAQ.vbp	omIM.vbp
omAU.vbp	omImp.vbp
omAudit.vbp	omIngestionManager.vbp
omBACS.vbp	omIngestionRules.vbp
omBankWizard.vbp	omInt.vbp
omBase.vbp	omKFIHelper.vbp
omBatch.vbp	omLC.vbp
omBC.vbp	omLockManager.vbp
omCaseTrack.vbp	omMP.vbp
omCC.vbp	omMQ.vbp
omCE.vbp	omOrg.vbp
omCF.vbp	omPack.vbp
omCK.vbp	omPAF.vbp
omCM.vbp	omPayProc.vbp
omCOG.vbp	omPDM.vbp
omComp.vbp	omPM.vbp
omCompRules.vbp	omPP.vbp
omCRM.vbp	omPrint.vbp
omCRMLock.vbp	omQASPro.vbp
omCRUD.vbp	omQQ.vbp
omCust.vbp	omQuest.vbp
omDataIngestion.vbp	omRB.vbp
omDataIngestionRules.vbp	omRC.vbp
omDM.vbp	omRegisterDocumentToCase.vbp
omROT.vbp	omTM.vbp
omRBlg.vbp	omToMSMQ.vbp

omToNOTHING.vbp
omToOMMQ.vbp
PrepareTemplateData.vbp

TemplateProcessor.vbp
TmBaseClient.vbp

5.7.2 Size

For estimation and cost purposes, source code has been divided into two groups: Client and Core.

The following tables show the projects that belong to each one of the groups and their respective total number of lines of code.

- Neither Blank lines nor comments are taken into account in the Effective Lines of code measurement.
- Information is taken from ArtinSoft's code analysis tools results.
- This information does not exclude the shared code used by the projects.

Client Lines of Code	
Lines	Total
Visual Lines	0
Code Lines	456,261
Comment Lines	108,905
Blank Lines	167,638
Total Effective Lines of Code	456,261

Table 1. Client Lines of Code

Client Files	
Files	Total
Classes	531
Modules	373
UserControls	0
PropertyPages	0
Forms	0
Total	904

Table 2. Client Files

Core Lines of Code	
Lines	Total
Visual Lines	300
Code Lines	160,580
Comment Lines	45,053
Blank Lines	18,305
Total Lines	160,880

Table 3. Core Lines of Code

Core Files	
Files	Total
Classes	177
Modules	272
UserControls	0
PropertyPages	0
Forms	3
Total	452

Table 4. Core Files

The following table shows the overall project size:

Overall Migration Project Size					
Priority Group	Visual Lines	Code Lines	Comment Lines	Blank Lines	Effective Lines of Code
Client	0	456,261	108,905	167,638	456,261
Core	300	160,580	45,053	18,305	160,880
Total	300	616,841	153,958	185,943	617,141

Table 5. Overall Project Size

Figure 2 shows how the Effective Lines of Code are distributed among the Client and Core project groups.

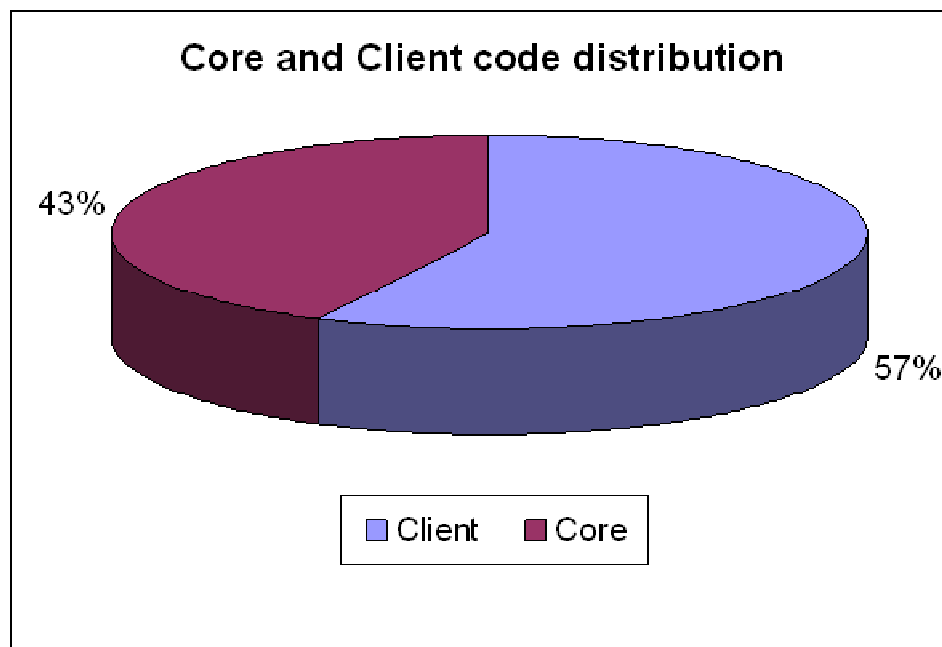


Figure 2. Client and Core Effective Lines of Code Distribution

5.7.3 Complexity

This section describes the complexity of the project for an eventual migration process.

Complexity Guidelines

The guidelines to consider the application as Low, Medium, High or Very High Complexity are:

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.

Because of the characteristics of the code and the low number of third party controls to be converted, complexity in this migration project can be considered as **Medium**.

5.8 Migration Strategy

Based on meetings that took place during the Ready, the objectives of the migration are:

- Achieve Functional Equivalence
- Ensure future evolution in .NET
- Guarantee strategic evolution for customers
- Obtain Microsoft Support
- Accomplish unification of shared code
- Take advantage of .NET features

The solutions to these goals are:

Goal	Proposed Solution
Functional Equivalence	Migrate to .NET
Continued evolution in .NET	Migrate to .NET and continue application evolution in .NET after reaching Functional Equivalence.
Strategic evolution for customers	Migrate to .NET and execute a Performance Testing and Tuning phase.

Unification of shared code	Migrate to .NET and merge and perform an analysis in the shared files to generate one or more assemblies.
Take advantage of .NET features	Migrate to .NET

Table 6. Migration goals and proposed solutions

5.9 Migration Requirements

In addition to the migration of Omega source code from Visual Basic 6.0 to C# .NET, Vertex has provided several migration requirements as explained in the next paragraphs.

5.9.1 Migration of Data Access Technology from ADO to ADO.NET

Data access based on ADO should be migrated to ADO.NET, using automatic conversion whenever possible.

5.9.2 Replacement of other COM controls and components, when possible

Other COM controls and components used by the Vertex source code would be converted to .NET native counterparts, if all the following conditions apply:

1. There is a sufficient number of occurrences of the component to justify an automatic or manual change as a part of the migration process, or it is impossible to keep the COM component working through COM Interoperability.
2. There is a .NET native equivalent for the COM component to be replaced.

5.9.3 MSXML to System.Xml

One of the most significant changes in this migration is moving all the previous MSXML objects and operations to the native System.XML namespace.

XML operations represent 90% of the code in COM+ components, this is why this change is highly important in terms of performance for Omega. The Visual Basic Upgrade Companion automatically migrates all the MSXML

elements to System.Xml native objects, with the exception of the MSXML2.FreeThreadedDOMDocument40.

ArtinSoft recommends including this object as part of the VBUC customization to reduce the manual effort that otherwise will be necessary to complete its migration.

5.9.4 Change the VB6 namespace functions to native .NET functions

VBUC currently minimizes the dependency of migrated code on the legacy VB6 library. However, ArtinSoft recommends to modify the VB6 source code to reduce the number of late-bound variables, which will increment the ability of VBUC to use native .NET Framework calls instead of legacy methods.

5.9.5 Use of Shared classes instead of Modules

C# .NET does not support the concept of Modules as in Visual Basic 6.0. All VB6 modules should be migrated to static classes (or classes in which all members are declared as static) to follow a more object-oriented coding style. Modules declared in the VB6 version of the source code would be converted to static classes, which will include the full qualification of method calls made to the new class.

5.9.6 Error handling conversion modifications

The Visual Basic Upgrade Companion automatically converts the most commonly used patterns of Unstructured Error Handling ("On Error GoTo") to Structured Error Handling ("Try..Catch"). Changes should be made for the specific case of Vertex, since the new exception model would replace error stack-tracing mechanisms currently used by Vertex. The following conversions will be automatically performed by VBUC.

Code that is currently converted to...	Must be converted to...
<pre>On Error GoTo FindListVbErr; { //C# Code 1 } FindListExit: { //C# Code 2 } FindListVbErr: //C# Code 3</pre>	<pre>try { //C# Code 1 } catch (System.Exception excep) { //C# Code 3 } Finally { //C# Code 2 }</pre>
<pre>string result = String.Empty; On Error GoTo ExitPoint; //C# Code 1</pre>	<pre>try { //C# Code 1 }</pre>

<pre> if (bSuccess) { xmlDoc = new XmlDocument(); if (xmlDoc == null) { throw new System.Exception(Constants.vbObjectError.ToString() + " " + gstrObjectName + "::" + strFunctionName + ", Failed to create XML Parser - check MSXML is installed, " + null + ", " + null); } //C# Code 1 foreach (PRINTER_DATA PrinterData_item in PrinterData) { //C# Code 2) } nodePrinters.AppendChild((XmlNode) nodePrinter); } On Error GoTo ExitPoint; //C# Code 2) } ExitPoint: //C# Code 3 </pre>	<pre> if (bSuccess) { xmlDoc = new XmlDocument(); if (xmlDoc == null) { throw new System.Exception(Constants.vbObjectError.ToString() + ", " + gstrObjectName + "::" + strFunctionName + " Failed to create XML Parser - check MSXML is installed, " + null + ", " + null); } //C# Code 1 foreach (PRINTER_DATA PrinterData_item in PrinterData) { //C# Code 2) } } catch (System.Exception excep) { } Finally { //C# Code 3 } </pre>
<pre> On Error GoTo BuildDbConnectionStringVbErr //C# Code 1) try { str Provider = (string) objWshShell.RegRead(strRegSection + gstrPROVIDER_KEY); } catch { } On Error GoTo BuildDbConnectionStringVbErr; if (Information.Err().Number != lngENTRYNOTFOUND && Information.Err().Number != 0) { throw new System.Exception(lngErrNo.ToString() + " " + strSource + " " + strDescription + " " + null + " " + null); } </pre>	<pre> try { //C# Code 1 try { s trProvider = (string) objWshShell.RegRead(strRegSection + gstrPROVIDER_KEY); } catch { } On Error GoTo BuildDbConnectionStringVbErr; if (Information.Err().Number != lngENTRYNOTFOUND && Information.Err().Number != 0) { throw new System.Exception(lngErrNo.ToString() + " " + strSource + " " + strDescription + " " + null + " " + null); } } </pre>

<pre>//C# Code 2) BuildDbConnectionStringVbErr: //C# Code 3</pre>	<pre>//C# Code 2) } } catch (System.Exception excep) { } Finally { //C# Code 3 }</pre>
<pre>const string cstrFunctionName = "RegisterCompensator"; Debug.WriteLine(cstrFunctionName); //If we have already registered the compensator then //exit the subroutine: if (bIsCompensatorRegistered) { return; } //Register the compensator with the DTC. //There is the possibility that the Compensator is in the //the process of recovering from a previous shutdown, hence //the error checking loop: On Error Resume Next; CrmLogControl.RegisterCompensator(COMPENSATOR_ProgID, COMPENSATOR_DESCRIPTION, (int) COMSVCSLib.tagCRMREGFLAGS.CRMREGFLAG_ALLPHASES); Application.DoEvents(); do { CrmLogControl.RegisterCompensator(COMPENSATOR_ProgID, COMPENSATOR_DESCRIPTION, (int) COMSVCSLib.tagCRMREGFLAGS.CRMREGFLAG_ALLPHASES); Application.DoEvents(); } while(Information.Err().Number == omCRMGlobal.XACT_E_RECOVERYINPROGRESS); //Was there an error registering the compensator? if (! (Information.Err().Number != 0)) {</pre>	<pre>const string cstrFunctionName = "RegisterCompensator"; Debug.WriteLine(cstrFunctionName); //If we have already registered the compensator then //exit the subroutine: if (bIsCompensatorRegistered) { return; } //Register the compensator with the DTC. //There is the possibility that the Compensator is in the //the process of recovering from a previous shutdown, hence //the error checking loop: try{ CrmLogControl.RegisterCompensator(COMPENSATOR_ProgID, COMPENSATOR_DESCRIPTION, (int) COMSVCSLib.tagCRMREGFLAGS.CRMREGFLAG_ALLPHASES); Application.DoEvents(); do { CrmLogControl.RegisterCompensator(COMPENSATOR_ProgID, COMPENSATOR_DESCRIPTION, (int) COMSVCSLib.tagCRMREGFLAGS.CRMREGFLAG_ALLPHASES); Application.DoEvents(); } while(Information.Err().Number == omCRMGlobal.XACT_E_RECOVERYINPROGRESS); //Was there an error registering the compensator? errAssistEx.errCheckError(cstrFunctionName, this.GetType().Name); } catch (System.Exception excep)</pre>

<pre> //Indicate that the compensator has been registered, so //subsequent calls to register the compensator abort: bIsCompensat orRegistered = true; return; } errAssistEx.errCheckError (cstrFunctionName, this.GetType().Name); </pre>	<pre> { bIsCompensatorRegistered = true; return; } </pre>
--	--

Table 7. Changes to the error handling conversion

Since these changes reduce code and allow exceptions to be thrown to the caller of a method without being handled, a performance loss is avoided. Figure 3 shows the proportion of error handlers that were translated to Structured Error Handling in the pre-migration of the Omega code done during the Ready analysis.

Error handlers that check for the Err.Number property in "If" or "Select Case" blocks will be marked with an EWI, (Error, Warning or Issue) as this occurrences will require additional coding to replace the Err.Number with a comparison against the type of the Exception that has been thrown.

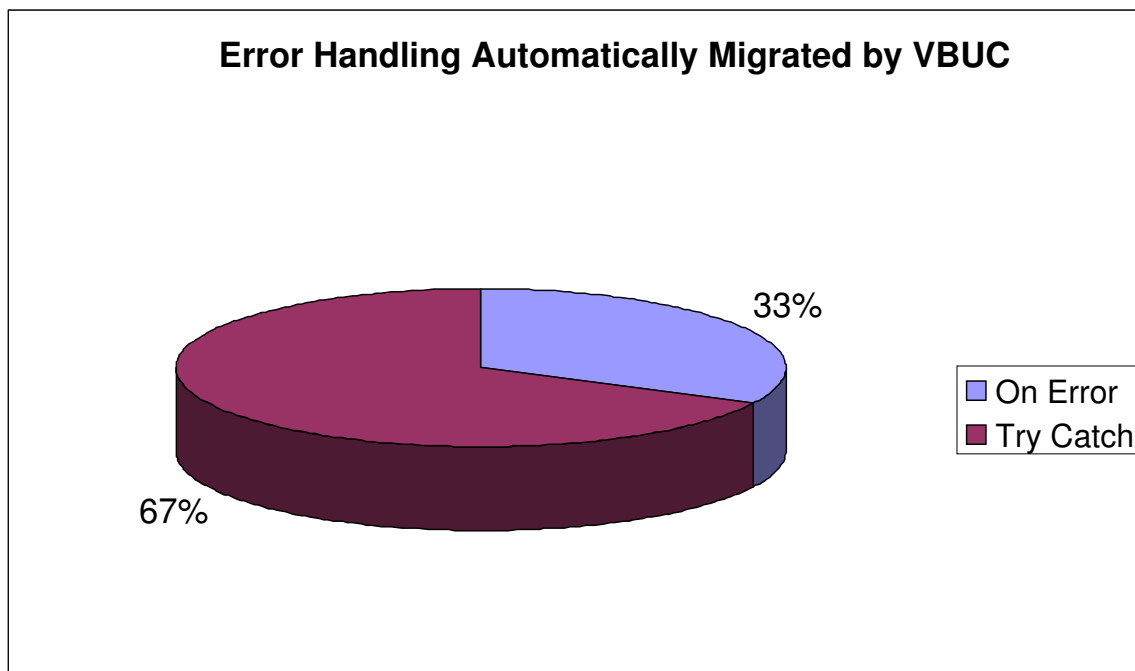


Figure 3. Proportion of Error Handlers converted to "Try..Catch"

6 Migration Proposal

The following is the description of the migration project for Vertex. This conversion will be performed following best practices and guidelines from ArtinSoft for migrations to the .NET platform.

6.1.1 Proposed Target Environment

Database	Microsoft SQL Server 2000/2005
Operating System	Windows 2003 Enterprise Server Service Pack 1
Development environment	Microsoft Visual Studio 2005
Web Server	Windows 2003 Enterprise Server Service Pack 1
Application Server	N/A

6.1.2 Target Architecture Description

1. Architecture in the migrated application will remain the same as in the original one, which is briefly described in section 5.3.

6.1.3 Use of Visual Basic Upgrade Companion

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

1. Coding conversions
 - 1.1. Modules to Shared classes (as shown in section 5.9.3)
 - 1.2. Error handling (as shown in section 5.9.4)
2. Additionally, the conversion of the following controls or libraries found in the Vertex code is already covered by the current version of Visual Basic Upgrade Companion:
 - 2.1. XmlParser 2 to System.Xml
 - 2.2. Scripting to System.IO

6.1.3.1 VBUC Customization Scope

This section shows the proposed sets of Properties, Methods and Events to be customized in Visual Basic Upgrade Companion in order to provide the most benefit to the migration process and its results, as well as to minimize manual effort necessary to reach Functional Equivalence.

Add COM Support

Adding COM Support for existing COM interface needs VBUC to be customized, following the next rules:

- For each ported class module, add to the top of the equivalent C# file:

```
using System.Runtime.InteropServices;
```

- Add the following attributes to the class:

```
[ComVisible(true)]  
[ProgId("[ProgId"])]  
[Guid("[Guid"])]  
public class [class Module Name]
```

The **ProgId** attribute should be the same as the VB6 component, i.e., the VB6 project name, followed by a ".", followed by the class module name. The **Guid** attribute should be the same as component CLSID. VBUC has to search inside the Registry using the ProgId value. After the key is found, the Guid value is located under the following registry path: HKEY_CLASSES_ROOT\[ProgId]\Clsid.

COM+ and Transactions

The Omega VB6 components are generally configured to run in COM+ and use COM+ transactioning. When a component is ported to .NET, several options are available:

1. Keep COM+ support: this is known as a "serviced component". This option is recommended when the .NET component needs to interoperate with existing VB6 components that are also running in COM+. The .NET and VB6 components can therefore enlist within the same distributed transactions in COM+, and the .NET components are installed into COM+ in the same way as existing VB6 components.

2. Replace COM+ with native .NET transactions in the **System.Transactions** namespace. This is the replacement for COM+ style transactions provided by Microsoft in .NET. It should be used for .NET components that do not need to interoperate with existing COM+ components. It is possible with this option for a .NET component to enlist in an existing COM+ transaction.
3. A combination of 1 and 2. A component can be provided with a serviced component class that wraps another class that uses native .NET transactioning.

The VBUC customization will apply option 1 due to the experience in this area with other migrations. The following list describes all procedures to be done by the VBUC tool.

1. Adding COM+ Support

1.1. Add System.EnterpriseServices

VBUC will add a reference to the **System.EnterpriseServices** assembly in the target project.

1.2. AssemblyInfo

VBUC will add the following lines to the **AssemblyInfo.cs** file:

```
using System.EnterpriseServices;

[assembly: ApplicationName("[ProjectName]")]
[assembly: ApplicationActivation(ActivationOption.Library)]
[assembly: ApplicationAccessControl(false)]
```

1.3. Registering in Component Services

After creating the dynamic library, it is required to register in Component Services using the *regsvcs* tool inside the .NET Framework v2.0.50727

1.4. Add COM+ Class Support

For each file where the **MTSTransactionMode** is not **Disabled** or **NotSupported** inside the VB6 Project, VBUC will perform the following steps:

- 1.4.1. Add the following lines to the top of the file:

```
using System.EnterpriseServices;
```

```
using System.Runtime.InteropServices;
```

- 1.4.2. Derive the class from **ServicedComponent**, and add a **Transaction** attribute, e.g.,

```
[ComVisible(true)]
[ProgId("[ProgId")]
[Guid("[Guid")]
[Transaction([.Net Transaction Option])]
public class [class Module Name] : ServicedComponent
```

The **TransactionOption** attribute should be set using the VB6 class module **MTSTransactionMode** property, as follows:

VB6 MTSTransactionMode	.NET TransactionOption attribute
0 – NotAnMTSObject	Disabled
1 – NoTransactions	NotSupported
2 – RequiresTransaction	Required
3 – UsesTransaction	Supported
4 – RequiresNewTransaction	RequiresNew

Table 8. MTSTransactionMode mappings

- 1.4.3. Comment out any declarations of GetObjectContext such as:

```
object GetObjectContext = null;
```

- 1.4.4. Map "GetObjectContext" to "ContextUtil" when SetComplete and SetAbort method are called.

Sample:

```
GetObjectContext.SetComplete maps to
ContextUtil.SetComplete
```

Interfaces

Convert interface classes to actual interfaces, and implement them explicitly in C#, e.g.:

```
void ICaseTrackingDO.GetCaseTrackingAllData(...)
```

AssemblyInfo.cs

The assembly info will be automatically generated by VBUC and will contain the following structure with blank spaces to be manually filled:

```
[assembly: AssemblyTitle("omCRUD")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("Visual Studio 2005")]
[assembly: AssemblyCompany("Vertex Financial Services")]
[assembly: AssemblyProduct("Omiga")]
[assembly: AssemblyCopyright("")]
[assembly: AssemblyTrademark("All rights reserved")]
[assembly: AssemblyCulture("")]
[assembly: ComVisible(True)]
[assembly: Guid("")]
[assembly: AssemblyVersion("")]
[assembly: ApplicationName("")]
```

1. The **AssemblyTitle** should be the same name as the project name.
2. The **AssemblyDescription** will be left blank.
3. The **AssemblyVersion** will be left blank.
4. The **Guid** must be the same as the CLSID for the Visual Basic 6.0 component.
5. The **AssemblyProduct** will be "Omiga".

Once the C# version of the component has been built, it will be possible to view its typelib information using OleView, by opening the .tlb file instead of the .dll file.

Porting Main Method

The VB6 **Main** method is automatically called when the component is created, allowing for component level initialization. To keep consistency in the code, ArtinSoft recommends to keep the Main method and call it from the constructor of the class.

Porting the App Object

There are a number of problems with porting of the VB App object, for example:

1. **App.Comments** may be replaced with null, generating compiler errors.
2. **App.Title** may be ported as is, which will generate compiler errors as the **App** type does not exist in .NET.
3. **App.Location** is ported as

```
System.Reflection.Assembly.GetExecutingAssembly().Location;
```

4. **App.LogEvent** is ported as is, but it will generate compiler errors as the **App** type does not exist in .NET.

To solve this, a static **App** class will be created and placed inside the **Vertex.Fsd.Omiga.VisualBasicPort** namespace, which will be used as a direct equivalent of the VB6 **App** object. The **App** class is static so that it can be used with the same syntax as the Visual Basic 6.0 App object, e.g., **App.Title**. Thus, it is possible to use the same syntax as in Visual Basic 6.0 but with the disadvantage of having to create a new **AppInstance** object every time one of the **App** properties is read. For performance reasons, it is recommended to consider replacing the calls to the **App** class in .NET with a single new **AppInstance** object, and using it directly.

Calls to the **App.LogEvent** method require the event log type (the second parameter) to be manually converted from VB6 to the appropriate **System.Diagnostics.EventLogEntryType** value, as follows

VB6 Constant	VB6 Value	System.Diagnostics .NET Value
vbLogEventTypeError	1	<code>EventLogEntryType.Error</code>
vbLogEventTypeWarning	2	<code>EventLogEntryType.Warning</code>
vbLogEventTypeInformation	4	<code>EventLogEntryType.Information</code>

Table 9. vbLogEventType Mappings

Rewrite Late Binding to COM Classes

There are two ways of porting late binding to COM Classes:

1. Using .NET Reflection in the System.Reflection namespace. *This is the preferred method.* It assumes that the component being called is a .NET component.
2. Calling the component via a late-bound COM call. It assumes that the component being called has a COM interface (e.g., a .NET component with a COM wrapper). *This method should not be used for .NET components for performance reasons.*

The following illustrates a late-bound COM call in .NET:

VB6 Code

```
Dim object As Object
Dim result As String
Set object = CreateObject(strProgId)
```

```
result = object.method(argument1, ..., argumentn)
```

C# code

```
Type objectType = Type.GetTypeFromProgID([ProgId], true);
object object = Activator.CreateInstance(objectType);

string result =
    Convert.ToString(
        objectType.InvokeMember(
            "[Method to call]",
            BindingFlags.Public |
            BindingFlags.InvokeMethod,
            null,
            object,
            new object[] { argument1, ..., argumentn }
        )
    );
```

Dispose and Using

If a type implements the **IDisposable** interface, VBUC will add a C# **using** construct to ensure that the object is disposed after being used.

For example, the method:

```
public static DataSet executeGetRecordSet(SqlCommand cmd)
{
    DataSet dataSet = new DataSet();

    SqlConnection conn = new
SqlConnection(adoGetDbConnectionString());
    cmd.Connection = conn;
    SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter(cmd);
    sqlDataAdapter.Fill(dataSet);

    return dataSet;
}
```

should be converted as:

```
public static DataSet executeGetRecordSet(SqlCommand cmd)
{
    DataSet dataSet = new DataSet();

    using (SqlConnection conn =
        new SqlConnection(adoGetDbConnectionString()))
    {
        cmd.Connection = conn;
    }
}
```

```

        using (SqlDataAdapter sqlDataAdapter =
            new SqlDataAdapter(cmd))
        {
            sqlDataAdapter.Fill(dataSet);
        }

    return dataSet;
}

```

When the **using** block exits (even if an exception is thrown), **Dispose** will be automatically called on the used object – this ensures that scarce resources (e.g., database connections) will always be freed.

The following items that are required by Vertex, are already supported by VBUC:

- Conversion of ParamArray Arguments.
- Correct porting of return keyword.
- Replacement of MSXML with System.Xml (The only object not supported is FreeThreadedDOMDocument40, which is listed above)
- Replacement of ADODB with ADO.NET (manual adjustments required)
- Replacement of Scripting.FileSystemObject with native .NET calls.
- Replacement of other COM Dependencies with native .NET calls.
- Partial conversion of Unstructured Error Handling to Structured Error Handling.
- Conversion of Enums to Integral Types.
- Conversion of ref and out Parameters
- Conversion of Optional Parameters
- Conversion of Local Variable Scope
- Handling of Loops Through Collections
- Handling of Redundant Foreach Variables
- Handling of "If not null" Conditions
- Handling of Integral Types
- Removal of Casting of Enums to Short or Int
- Usage of the best overloaded method match for 'Microsoft.VisualBasic.ErrObject.Raise(int, object, object, object, object)' without invalid arguments
- Usage of 'adoCRUDGetSchemaRefNode' as a variable instead of a method

- Solution of "Argument '1': cannot convert from 'byte' to 'byte[]'" error message.
- Solution of "Argument '1': cannot convert from 'ref int' to 'out object'" error message.
- Solution of "Cannot implicitly convert type typeA to typeB. An explicit conversion exists (are you missing a cast?)" error message.
- Solution of "Cannot implicitly convert type 'string' to 'ADODB.Connection'" error message.
- Solution "No overload for method 'ReadText' takes '0' arguments" error message.
- Solution of "No such label 'OmRequestExit' within the scope of the goto statement" error message.
- Solution of "The name 'xmlSchemaNodeList' does not exist in the current context" error message.

For tracking purposes, VBUC will write an EWI (Error, Warning or Issue) in the migrated C# code each time an unsupported class, property, method or event is found from the customized areas. Because of this, the total number of EWIs found in the generated C# code after customization may increase.

6.1.4 Items to be addressed from Visual Basic 6.0

6.1.4.1 Remove occurrences of Unstructured Error Handling that have no equivalent in .NET

Whenever possible, the following error-handling patterns should be rewritten in VB6 using the standard "On Error GoTo [Handler]" format to increase the percentage of automatic conversion to Structured Error Handling (Try..Catch).

1. On Error Resume
2. On Error Resume Next
3. On Error GoTo 0

6.1.4.2 Execute the Code Advisor tool and implement recommended fixes

The Code Advisor is a pre-migration diagnostic tool that can be freely downloaded from Microsoft at the following URL:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=a656371a-b5c0-4d40-b015-0caa02634fae&DisplayLang=en>

The Code Advisor installs itself as an Add-In to the Visual Basic 6.0 development environment, and will generate a set of suggestions on how to modify VB6 source code in order to make it easier to migrate to .NET. These suggestions are written as comment lines within the Visual Basic 6.0 source code.

Some of the changes suggested by the Code Advisor are already addressed automatically by the Visual Basic Upgrade Companion. For a list of these changes, refer to the following article on the ArtinSoft web site:

<http://www.artinsoft.com/VB-Upgrade-Companion-vs-CodeAdvisor.aspx>

The following table shows the list of Code Advisor suggestions (also called "FixIts") as obtained from a sample of the Vertex source code.

Description
Late Binding of Variant or Object
Missing Option Explicit
Variant-Returning String Function
Property/Method/Events Not Upgraded
API Declare statements that include parameters typed using 'As Any' will not be upgraded.
When a #If condition evaluates to False, the #If...#End If block is not upgraded. The Upgrade Wizard does not reliably evaluate whether #If conditions are True or False.
In Visual Basic .NET, On ... GoTo is not supported.

Table 10. Code Advisor "FixIts" found on Omega code

6.1.4.3 Create a Common DLL

In order to avoid redundant code, Artinsoft recommends creating a DLL or a group of DLLs with the shared code. This will help organize the code in logical libraries.

6.2 Project Structure & Responsibilities

The following figure shows the stages of the migration process for Vertex:

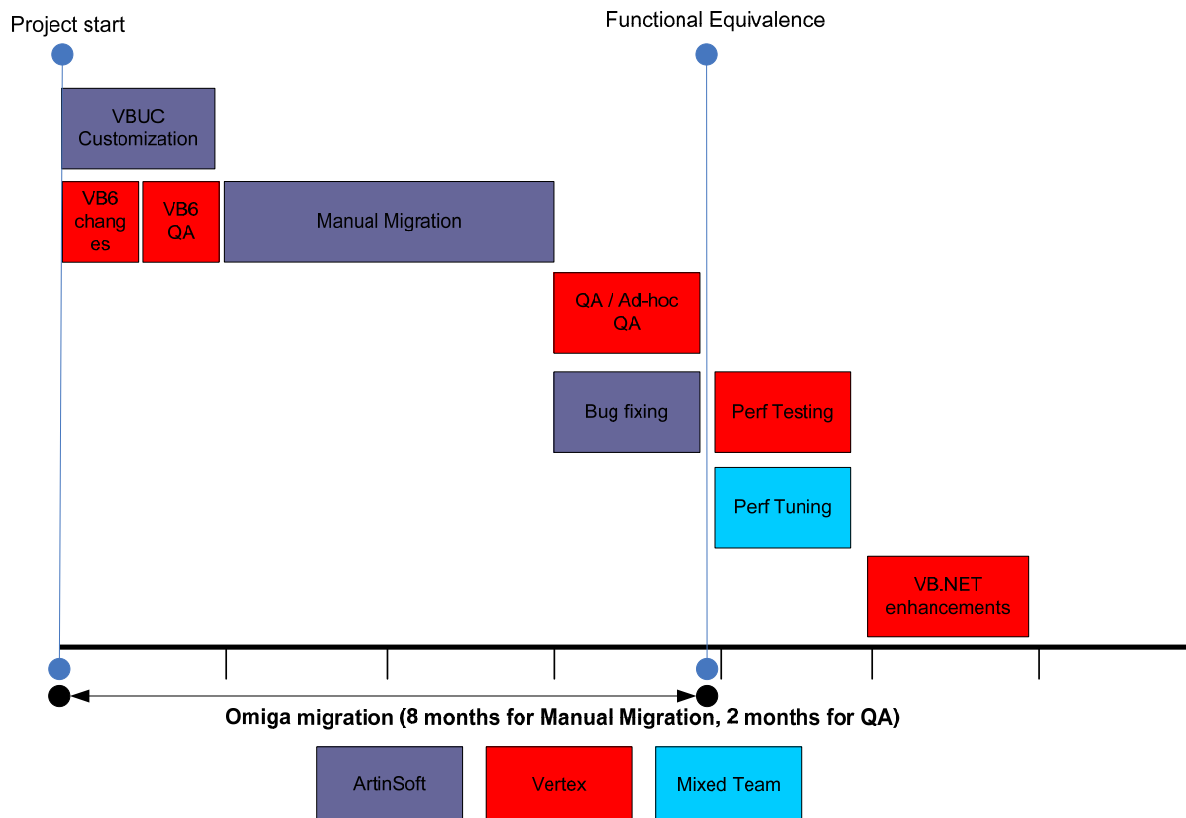


Figure 4. **High-level Project Layout (Migration Plan 1)**

In Migration Plan 1, a standard ArtinSoft team will be used. In Migration Plan 2, more resources will be added to compress the Manual Migration phase, producing a duration of 3 calendar-months for Manual Migration and Developer Testing together.

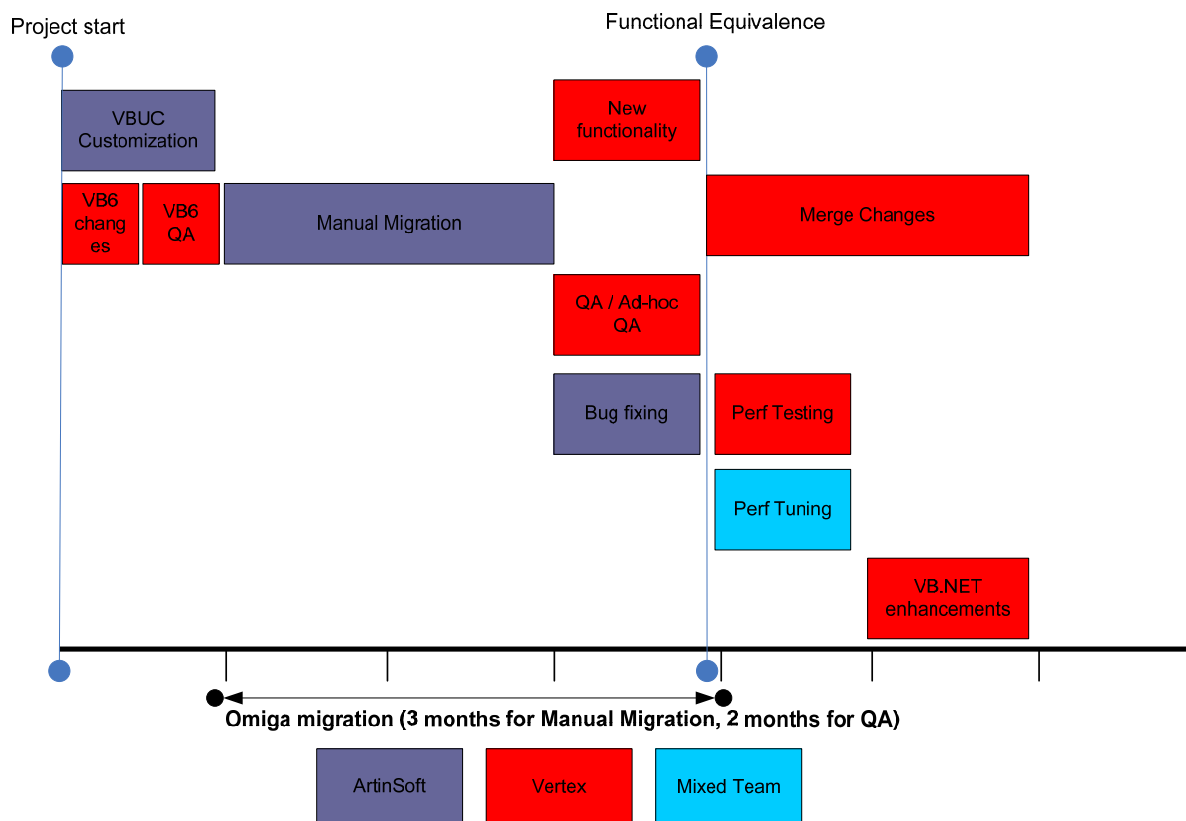


Figure 5. **High-level Project Layout (Migration Plan 2)**

Activities shown in purple boxes will be performed by ArtinSoft, while red box activities will be performed by Vertex. Activities shown in light blue boxes will be performed by a Mixed Team.

ArtinSoft will adapt its standard migration process to fit the requirements provided by Vertex. A description of each of the tasks that have been identified as part of this migration project is provided in the following paragraphs.

6.2.1 Visual Basic Upgrade Companion Customization (VBUCC)

During this phase, ArtinSoft will implement migration rules to cover the automation scope defined in section 6.1.3: Use of Visual Basic Upgrade Companion.

6.2.1.1 Setup

Source code installation and project kickoff.

6.2.1.2 Kickoff

A kickoff conference call would take place with both ArtinSoft and Vertex personnel involved in the project. The following topics should be covered in this initial project meeting:

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

6.2.1.3 Installation and Configuration

Machine installation and configuration; source code and components installation. Migration machines should duplicate a Vertex development configuration. On these migration machines, regular migrations will be performed to enable tracking of Customization progress.

6.2.1.4 Specification

Mappings for the coding requirements shown in section 6.1.3 will be specified and documented. These specifications must be approved by Vertex before they are implemented.

6.2.1.5 Implementation

Mappings specified in the previous step will be implemented according to that specification.

6.2.1.6 Testing

ArtinSoft QA will test the implemented mappings with Vertex code against the specification produced in 6.2.1.4.

6.2.1.7 Code Verification

Migrate the Omega VB6 projects shown in section 5.7.1 and verify that the Visual Basic Upgrade Companion results are as expected. This will create a new migration errors and issues baseline for the migration project.

6.2.2 VB6 Changes and QA

During this phase, Vertex will implement the requirements listed in section 6.1.4 in Visual Basic 6.0. Also, Vertex will implement new functionality in Omega. ArtinSoft strongly recommends dedicating sufficient time to testing (QA) new and modified code and fixing bugs that are found during this process. This will ensure that the Migration phase of the project is based on code that has been tested and verified as an input, minimizing the risk of porting functional bugs to .NET.

6.2.3 Migration

Automatic and manual conversion tasks, as described below.

6.2.3.1 Kick-off and Migration Training

Before the manual migration process begins, it is recommended to have a Project Kickoff with the entire Project Team, to cover aspects such as project organization, roles, objectives and calendar, acceptance criteria, communication channels and follow-up mechanisms. Also, an ArtinSoft consultant will provide a Migration Training for the project team, to reduce learning curves and increase productivity. This process would take one calendar-week.

6.2.3.2 Setup

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

6.2.3.3 Automatic Migration

Execute the customized version of Visual Basic Upgrade Companion to generate Green Code in C# .NET.

6.2.3.4 Manual Migration

Manual conversion tasks, as described below.

6.2.3.4.1 For Each Project:

Automatic Migration

- **Automatic Code Translation:** migration of the .vbp project, after source code preparation.

Coding

- **EWI Resolution and Compilation Errors:** resolve existing compilation errors in the C# code that may result from the migration process and resolve pending migration issues; upgrade pending RDO, ADO and third-party code.
- **Runtime Bug Fixing:** run the application and fix run-time errors.

6.2.3.5 Developer Testing

After the .NET version of Omega is generated, Developer Testing begins. During this phase, the developers involved in the migration process test the migrated and integrated 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.

6.2.3.6 Administrative

Administrative work that is necessary for the Migration phase of the project, such as technical and follow up meetings.

6.2.3.7 Configuration Management

Configuration Management (CM) and code backup tasks.

6.2.3.8 Project Management

Project Management tasks such as:

- Project planning
- Requirements tracking
- Scope and budget control
- Risks management

6.2.4 Quality Assurance (QA)

Quality Assurance activities will certify that the application behavior is as expected. This first Quality Assurance phase will be executed with a basic set of test cases created to verify the core functionality of Vertex.

6.2.5 Ad-hoc Quality Assurance (Ad-hoc QA)

During this task, a group of testers will interact with the migrated application from an end-user standpoint, documenting functional bugs and reporting them to the Development team.

6.2.6 Bug Fixing

During this task, the project team will fix bugs reported by the QA team as a result of the QA tasks.

6.2.7 Performance Testing (Perf. Testing)

For this task, side by side execution of the same application transactions on different machines will be performed by the testing team. The objective is to identify transactions for which the migrated application may have significant performance degradation when compared to the original. Detected performance issues would be documented and reported to the Development team to be studied and resolved. Machines used in this testing process must have the same hardware configurations. More precise timing on a particular issue could be performed using the Trace and Debug classes that are included in the .NET framework if desired.

6.2.8 Performance Tuning (Perf. Tuning)

During this task, the project team will resolve performance issues reported by the Performance Testing team as a result of its testing tasks.

6.2.9 C# Enhancements (C#)

After Functional Equivalence has been achieved and performance has been verified, Vertex will implement new functionality in Omega according to the Omega product roadmap. Examples of the expected enhancements are:

1. Make changes to critical applications where Vertex has identified performance problems.
2. New requirements from customers.

6.3 Business Case

The migration of Omega to C# .NET will increase technological advantage over competing products. This will eventually strengthen the presence of Vertex in the market. Also, it will provide more possibilities to enhance the application and implement new features in .NET.

Porting Omega to .NET will also ensure continued support from Microsoft for this mission-critical application.

6.4 Final Recommendation

The recommended course of action is as follows:

1. Extend the Visual Basic Upgrade Companion to increase automatic conversion of the following areas, according to the scope defined in section 6.1.3.
2. Take advantage of the support already present in VBUC for the migration of the following areas:
 - Scripting to System.IO
 - MSXML to System.Xml
 - Unstructured Error Handling to Structured Error Handling
 - Visual Basic Functions
3. Execute the Visual Basic 6.0 code modifications listed in section 6.1.4. These modifications can be done in parallel with the Visual Basic Upgrade Companion Customization process, and will simplify the manual changes necessary to reach Functional Equivalence in C# .NET.
 - 3.1. Execute a thorough Quality Assurance (QA) process after making these modifications to the VB6 code. This will ensure that migration is performed on a code base that has been verified, minimizing the risk of porting existing bugs to the new platform.
4. Migrate the 77 Visual Basic 6.0 projects with the customized version of Visual Basic Upgrade Companion, then apply manual changes to complete the process, in the following order:

- 4.1. Fix compilation errors.
- 4.2. Fix upgrade Warnings, Errors and Issues (EWIs).
- 4.3. Fix runtime issues in Developer Testing.

There are 2 strategies that ArtinSoft recommends for the migration of Omega. Strategy 1 is as follows:

- a. Migrate and generate one or more libraries for the source code that is common to the 77 projects that will be migrated.
- b. Migrate the client projects using a reference to a common DLL.
- c. Migrate the core projects using a reference to a common DLL.

Strategy 2 is as follows:

- a. Migrate and generate one or more libraries for the source code that is common to the 77 projects that will be migrated.
- b. Migrate the client projects using a reference to a common DLL.
- c. Migrate the core projects using a reference to a common DLL.
- d. Vertex will add new functionality to a copy of client and core project.
- e. Once Functional Equivalence has been achieved, Vertex will merge those changes to a definite version.

ArtinSoft considers strategy #1 to be the best option, and in the following sections of this document, time estimation and budget for this option are presented. These numbers will have to be reviewed and adjusted if Vertex decides to go with the second strategy.

- 5. Parallel to migration (point 4); develop a set of Test Cases that will be used to test the functionality of the applications being migrated.
- 6. Execute a Quality Assurance (QA) phase based on the set of test cases previously created by Vertex. This will help ensure that the core functionality of the migrated application works as expected. In parallel, a Bug Fixing task will be executed to resolve issues found during this QA phase.
- 7. After finishing the first QA phase, an ad-hoc Quality Assurance phase will be performed, delimited by duration. This will be accomplished by having a group of testers interact with the application as end users and document the issues found. Developers would fix the reported bugs in parallel with this task.
- 8. Execute a Performance Testing and Tuning phase. This will be achieved by having developers compare the VB6 and C# applications side by side on different machines with the same hardware configuration. Points in which the migrated application performs significantly slower than the original

one will be documented by the QA team to be reviewed and improved by the Development team.

6.5 Additional recommendations

As stated by Vertex, code modifications and development in VB6 will continue before the Migration phase of the project begins. ArtinSoft does not expect these additional changes to the VB6 code base to increase the migration complexity of the source code. The following list provides several key recommendations regarding how this source code should be written to minimize manual changes necessary to migrate the code to .NET.

1. Avoid late binding: undeclared variables and variables declared "As Variant" will decrease the level of automatic migration. Always use "Option Explicit" at the beginning of each code file.
2. Do not write new unstructured error handlers using "On Error Resume", "On Error Resume Next" or "On Error GoTo 0".
3. Avoid adding new third-party COM controls to the current Visual Basic 6.0 code base.
4. Create one or more libraries in .NET with the shared code to centralize the code and organize it logically to accomplish an object oriented structure.

6.6 Time Estimation

The purpose of this section is to provide an estimation of the time needed to execute the different phases of this migration project.

6.6.1 Visual Basic Upgrade Companion Customization

ArtinSoft will customize the Visual Basic Upgrade Companion according to the scope described in section 6.1.3 in approximately 1.5 calendar-months.

Requirement	Duration (calendar-months)
All Coding Conversions (as presented in section 6.1.3)	1.5
Total	1.5

Table 11. VBUC Customization Duration

6.6.2 Migration

The estimation for the Migration phase of this project was calculated utilizing ArtinSoft's Code Analysis tools and its past experience on similar projects. Productivity for each portion was adjusted according to the Omega complexity.

The duration estimation provided here is based upon a Mixed Team of ArtinSoft consultants and Vertex testers. Based on the calculated productivity levels, duration estimation is as follows (measured in calendar-months):

Migration Plan	Duration
Migration Plan 1	10
Migration and Developer Testing	8
Testing	2
Migration Plan 2	5
Migration and Developer Testing	3
Testing	2

Table 12. Project Duration According to Migration Plans 1 and 2

6.7 Anticipated Project Budget

This section shows the anticipated budgets for each one of the elements involved in the migration of the Omega application. All prices are shown in U.S. Dollars.

6.7.1 Budget for Visual Basic Upgrade Companion License

ArtinSoft proposes to license the Visual Basic Upgrade Companion for the migration of the Omega application at a fee of \$89,999.

6.7.2 Budget for Visual Basic Upgrade Companion Customization

ArtinSoft proposes to meet the VBUC Customization requirements shown in section 6.1.3, with the prices presented in the following table.

Requirement	Price
Add COM Support	\$3,847
COM+ and Transactions	\$7,695
Interfaces	\$2,308
AssemblyInfo.cs	\$3,463
Porting Main Procedure	\$770
Porting the App Object	\$1,539
Rewrite Late Binding to COM Classes	\$1,539
Incorrect Porting of VB Function Return (delete objXmlDoc = null)	\$192
Replace MSXML (map parseError to try..catch(XmlException)	\$192
Stored Procedure Parameters	\$770
Replace Microsoft.VisualBasic (Map Information.IsNumeric)	\$385
Remove VSA Support	\$770
Dispose and Using	\$2,309
Cannot implicitly convert type 'uint' to 'int'. An explicit conversion exists (are you missing a cast?)	\$95
'CompareMethod' is an ambiguous reference between 'Microsoft.VisualBasic.CompareMethod' and 'Scripting.CompareMethod'" Manual Changes	\$770
cmd.Execute , , adExecuteStream. Argument '1': cannot convert from '<null>' to 'out object'	\$385
The variable 'variable' is assigned but its value is never used	\$770
Total	\$27,798

Table 13. VBUC Customization budget

6.7.3 Budget for Migration

The following table shows the anticipated budget for the execution of the migration project as presented in section 6.2. Separate estimates are shown for Migration Plan 1 and 2.

Migration Plan	Price
Migration Plan 1 (Manual Migration and Bug Fixing according to section 6.2 and Figure 4)	\$481,003
Migration Plan 2 (Manual Migration and Bug Fixing according to section 6.2 and Figure 5)	\$582,103

Table 14. Anticipated Budget for Migration, according to Migration Plans 1 and 2

6.7.4 Overall Migration Budget

The following table shows the overall migration budget for each one of the migration plans, including VBUC License and Customization. The license price assumes that all the customizations shown in section 6.7.2 are made.

Migration Plan	Migration Plan 1	Migration Plan 2
VBUC License	\$89,999	\$89,999
VBUC Customization	\$27,798	\$27,798
Migration Project	\$481,003	\$582,103
Total	\$598,800	\$699,900

Table 15. Overall Migration Budget, according to Migration Plans 1 and 2


6.8 Appendix A: Client Statistics

REFERENCES

ACTIVEX REFERENCES

Name	CLSID	Version	Path
OLE Automation	00020430-0000-0000-C000-000000000046	2.0#0	..\..\..\..\WINDOWS\System32\stdole2.tlb
Microsoft XML, v4.0	F5078F18-C551-11D3-89B9-0000F81FE221	4.0#0	..\..\..\..\WINDOWS\system32\msxml4.dll
Microsoft Transaction Server Type Library	74C08640-CEDB-11CF-8B49-00AA00B8A790	1.0#0	..\..\..\..\WINDOWS\system32\comsvcs.dll
Microsoft ActiveX Data Objects 2.7 Library	EF53050B-882E-4776-B643-EDA472E8E3F2	2.7#0	..\..\..\..\Program Files\Common Files\System\ado\msado27.tlb
Omiga to Message Queue Type Library	EB11E2D1-3B22-11D5-94E3-00104B3FBCCB	1.0#0	..\..\..\..\Build Epsom2\2 Int Code\ExternalComponents\InstallableTypeLibraries\OmigaToMessageQueueInterface.tlb
Windows Scripting Host Object Model (Ver 1.0)	F935DC20-1CF0-11D0-ADB9-00C04FD58A0B	1.0#0	..\..\..\..\WINDOWS\System32\wshom.ocx
Microsoft Scripting Runtime	420B2830-E718-11CF-893D-00A0C9054228	1.0#0	..\..\..\..\WINDOWS\System32\scrrun.dll
omBase	10E98D81-E223-4674-BD4E-369841AF4C71	1.2#0	..\..\..\Core\code\Base\omBase\omBase.dll
omTP	ACD64FFD-6449-4F6B-BF2D-2897B697608C	1.0#0	..\..\ThirdParty\omTP\omTP.dll
omApp	A82515BC-5DBD-4B4E-B20D-482F8CDA5301	1.0#0	..\..\Application\omApp\omApp.dll
omAQ	E3236E77-B2CB-4C40-9AFE-5AFB84202A12	1.0#0	..\..\ApplicationQuote\omAQ\omAQ.dll
omCE	3677667A-D080-43EB-B9E2-5F9193D098AC	1.0#0	..\..\CustomerEmployment\omCE\omCE.dll
omCF	67685788-6394-465F-B16D-0C7227753985	1.0#0	..\..\CustomerFinancial\omCF\omCF.dll
omCust	8BE45F91-9E06-4D97-9216-1B033AF2CAD6	1.0#0	..\..\Customer\omCust\omCust.dll
omCC	B9F22B67-1C04-4407-	1.0#0	..\..\CreditCheck\omCC\om

	8E77-60014777F6DF		CC.dll
omIC	22B92477-2651-4428-862A-B35375567CF3	1.0#0	..\..\Income Calcs\omIC\omIC.dll
omQQ	3B0F70E0-5A8B-43BE-A275-3867EE1FEB7F	1.0#0	..\..\QuickQuote\omQQ\omQQ.dll
Shared Property Manager Type Library	2A005C00-A5DE-11CF-9E66-00AA00A3F464	1.0#0	..\..\..\..\..\WINDOWS\System32\COMSVCS.DLL
omOrg	5492D917-89EC-408F-8692-73F0E9A6E21A	1.0#0	..\..\..\Core\code\Organisa tion\omOrg\omOrg.dll
omAU	355A1E4B-CB82-445F-948E-448F336E1499	1.1#0	..\..\..\Core\code\Audit\om AU\omAU.dll
omBankWizardW rapper	D96FE868-B4B7-4F26-A9BC-F260FCF8E7B3	1.0#0	..\..\..\Core\code\BankWiz ard\omBankWizard\omBan kWizard.exe
omIM	DA68D365-4BF5-4A0C-80AE-47DE17755CC4	1.0#0	..\..\Intermediary\omIM\omIM.dll
MessageQueueCo mponentVC 1.0 Type Library	B0C456C2-4DAE-11D4-823C-005004E8D1A7	1.0#0	..\..\..\Core\bin\VC\Messag eQueue\Windows2000\Mes sageQueueComponentVC.d ll
OmPrint	D139DF12-6520-4ACC-B529-2D96E0FCC07D	1.0#0	..\..\..\Core\code\Printing\ omPrint\omPrint.dll
omRB	646E8612-6F26-40A0-BC90-3B53C7C4E2B6	1.0#0	..\..\..\Core\code\RequestB roker\omRB\omRB.dll
omAPRules	A62A9C43-DF6D-4326-8C23-3C63EC4E6942	1.0#0	..\omAPRules\omAPRules.d ll
omBatch	CB10DD51-E375-4020-B218-05D8DD76F476	1.0#0	..\..\BatchProcessing\omBa tch\omBatch.dll
omCM	2D5F231A-EAF6-4012-8C27-08C1C5DE08FF	1.0#0	..\..\CostModelling\omCM\ omCM.dll
omMP	694619C1-3B66-4B3D-B32B-6A9F982A88A6	1.0#0	..\..\MortgageProduct\omM P\omMP.dll
omRA	972B7427-4164-4F3E-B440-0180233C07D2	1.0#0	..\..\RiskAssessment\omRA \omRA.dll
omTM	A9880657-A312-4D46-9C06-DCEF77007E2F	1.0#0	..\..\TaskManagement\omT M\omTm.dll
omBC	FC24365C-4233-4CC8-A5A8-3CBEFAB36C08	1.0#0	..\..\BuildingsAndContents\ omBC\omBC.dll
omLC	F110B99B-B30F-4E44-8D29-460F42376EE0	1.0#0	..\..\LifeCover\omLC\omLC .dll
omPayProc	937A72DE-F244-4B88-8D75-6982598E82D3	1.0#0	..\..\PaymentProcessing\om PayProc\omPayProc.dll
omPP	DC3076C7-9643-48E7-88BA-44F31E1B92EB	1.0#0	..\..\PaymentProtection\om PP\omPP.dll
omCRUD	39DC2E58-E95D-408B-90D7-F9BBC7502FC8	1.1#0	..\..\..\Core\code\CRUD\om CRUD\omCRUD.dll
COM+ Services Type Library	2A005C00-A5DE-11CF-9E66-00AA00A3F464	1.0#0	..\..\..\..\..\WINDOWS\System32\COMSVCS.DLL
omCRMLock	291EA7E2-BD8F-482C-9192-2F2E60E15880	1.0#0	..\..\..\Core\code\Compens atingResourceManager\om CRMLock\omCRMLock.dll

	Vertex	Technical Analysis & Migration Proposal
---	--------	---

PolarisInterface 1.0 Type Library	A5648C33-A7D3-11D3-92C1-0060087A1EBE	1.0#0	..\..\..\..\..\Build Epsom2\2 Int Code\ExternalComponents\TypeLibraries\PolarisInterface.tlb
AlphaCOMPlus 1.0 Type Library	C815B99D-A83B-400B-9663-A06DFB93A940	1.0#0	..\..\..\Core\code\AlphaPlu s\Components\Dev\AlphaC OMPlus.dll
eKFI	58C02F95-6D43-4D65-B951-4CBDF3C4D48A	d.0#0	..\..\..\Core\code\KFI PRINT SOLUTION\eKFI\eKFI.dll
omDPS	00984230-4E44-4F70-9F7E-ECC9ADCEF1E5	2.1#0	..\..\..\Core\code\Printing\ omDPS\omDPS.dll
omPDM	AAEF6A04-D7B1-43CE-AEFD-B341BBA3BD9C	1.0#0	..\..\Print Data Manager\omPDM\omPDM.d ll
KFI Helper Component	F0426951-844D-4463-A27E-AF45CA17311D	1.0#0	..\..\KeyFactsIllustration\o mKFIHelper\omKFIHelper.d ll
omPack	07597F4D-A77E-41C7-86D6-335A1D3110B8	1.0#0	..\..\Pack Manager\omPack\omPack. dll
Microsoft Data Environment Instance 1.0	3D5C6BF0-69A3-11D0-B393-00A0C9055D8E	1.0#0	..\..\..\..\..\Program Files\Common Files\designer\MSDERUN.D LL
omEXP	509B1FC3-713A-4B7F-91EE-B29BF9D776EA	1.0#0	..\omExp\omEXP.dll
ExperianInterface 1.0 Type Library	B55B169E-D97F-11D3-AD7E-005004E8D1A7	1.0#0	..\..\..\..\..\Build Epsom2\2 Int Code\ExternalComponents\TypeLibraries\ExperianInte rface.tlb
omAdmin	31DF831E-B6E0-4B4A-9E98-42A9D7A6326E	1.0#0	..\..\Administration Interface\omAdmin\omAd min.dll
om4to3	8DE80F91-F920-4519-BC7C-A4F401D2EF58	1.0#0	..\..\Omega4ToOmega3Dow nload\om4to3\om4To3.dll
omSub	C5F9A5F0-CD03-4DEA-A241-7D39749197F2	1.0#0	..\..\Submission\omSub\o mSub.dll
MsgTm	C4749ED1-9089-4D6B-9B63-CB5E1E4557AF	1.0#0	..\..\TaskManagement\msg TM\MsgTm.dll
Microsoft ActiveX Data Objects 2.6 Library	00000206-0000-0010-8000-00AA006D2EA4	2.6#0	..\..\..\..\..\Program Files\Common Files\system\ado\msado26 .tlb
ODIConverter 1.0 Type Library	12A83E98-ACDD-4FF9-87A8-F895DD747E07	1.0#0	..\..\..\..\..\Build Epsom2\2 Int Code\ExternalComponents\TypeLibraries\ODIConverte r.tlb
MSGComp 1.0	70D800F1-58A3-11D4-	1.0#0	..\..\..\..\..\Build

Type Library	8244-000102125FBA		Epsom2\2 Int Code\ExternalComponents\TypeLibraries\MSGComp.tlb
omPM	2772CC24-7C61-4A82-A158-D6284D6F383F	1.2#0	..\..\..\Core\code\Printing\omPM\omPM.dll
PrepareTemplate	864E60B2-1612-47C3-B760-C7A8798E71CE	1.1#0	..\..\..\Core\code\Printing\PrepareTemplate\PrepareTemplateData.dll
Windows Script Host Object Model (Ver 1.0)	F935DC20-1CF0-11D0-ADB9-00C04FD58A0B	1.0#0	..\..\..\..\..\WINDOWS\System32\wshom.ocx
omAppProc	C6C8CF1B-67FB-48AE-9391-9E2A5E0B96F0	1.0#0	..\..\ApplicationProcessing\omAppProc\omAppProc.dll
	AE9991E6-DF5B-4A98-946D-1162DBAFAF62	1.2#0	MsgTm.dll
Microsoft XML, v3.0	F5078F18-C551-11D3-89B9-0000F81FE221	3.0#0	..\..\..\..\..\WINDOWS\System32\msxml3.dll
OmRARules	9787F2A0-4BAF-4F62-ADE5-0239BAC45BB6	1.0#0	..\..\RiskAssessment\omRaRules\omRARules.dll

WINDOWS API DECLARATIONS

Name	Lib
WinCoCreateGuid	OLE32.DLL
RegCloseKey	advapi32.dll
RegCreateKey	advapi32.dll
RegDeleteKey	advapi32.dll
RegDeleteValue	advapi32.dll
RegOpenKey	advapi32.dll
RegQueryValueEx	advapi32.dll
RegSetValueEx	advapi32.dll
SetTimer	user32
KillTimer	user32
RegCreateKeyEx	advapi32.dll
RegOpenKeyEx	advapi32.dll
RegQueryValueExString	advapi32.dll
RegQueryValueExLong	advapi32.dll
RegQueryValueExNULL	advapi32.dll
RegSetValueExString	advapi32.dll
RegSetValueExLong	advapi32.dll
SHDeleteKey	Shlwapi.dll
ShellExecute	shell32.dll
GetDesktopWindow	user32
VSAWinCoCreateGuid	OLE32.DLL
	advapi32

6.9 Appendix B: Core Statistics

REFERENCES

ACTIVEX REFERENCES

Name	CLSID	Version	Path
OLE Automation	00020430-0000-0000-C000-000000000046	2.0#0	C:\WINDOWS\System32\st dole2.tlb
Microsoft Transaction Server Type Library	74C08640-CEDB-11CF-8B49-00AA00B8A790	1.0#0	C:\WINDOWS\system32\co msvcs.dll
Omiga to Message Queue Type Library	EB11E2D1-3B22-11D5-94E3-00104B3FBCCB	1.0#0	..\..\ExternalComponents\I ninstallableTypeLibraries\Om igaToMessageQueueInterfa ce.tlb
Microsoft XML, v4.0	F5078F18-C551-11D3-89B9-0000F81FE221	4.0#0	C:\WINDOWS\system32\m sxml4.dll
Microsoft ActiveX Data Objects 2.7 Library	EF53050B-882E-4776-B643-EDA472E8E3F2	2.7#0	C:\Program Files\Common Files\System\ado\msado27 .tlb
COM+ Services Type Library	2A005C00-A5DE-11CF-9E66-00AA00A3F464	1.0#0	..\..\..\..\..\WINDO WS\System32\COMSVCS.D LL
	10E98D81-E223-4674-BD4E-369841AF4C71	1.2#0	..\..\Base\omBase\omBase .dll
object safety 1.0 Type Library	60DB7853-08E4-11D5-B710-0060087A1E40	1.0#0	ObjectSafety.tlb
Microsoft XML, v3.0	F5078F18-C551-11D3-89B9-0000F81FE221	3.0#0	..\..\..\..\..\WINDO WS\System32\msxml3.dll
dmsCompression 1.0 Type Library	85EEB9C0-CB35-46DE-A282-6E112AE5D9D4	1.0#0	..\..\..\bin\VC\dmscompres sion.dll
Microsoft Scripting Runtime	420B2830-E718-11CF-893D-00A0C9054228	1.0#0	..\..\..\..\..\WINDO WS\System32\scrrun.dll
Windows Scripting Host Object Model (Ver 1.0)	F935DC20-1CF0-11D0-ADB9-00C04FD58A0B	1.0#0	C:\WINDOWS\System32\w shom.ocx
AlphaCOMPlus 1.0 Type Library	C815B99D-A83B-400B-9663-A06DFB93A940	1.0#0	..\..\AlphaPlus\Component s\Dev\AlphaCOMPlus.dll
Microsoft ActiveX Data Objects 2.6 Library	00000206-0000-0010-8000-00AA006D2EA4	2.6#0	..\..\..\..\..\Progr am Files\Common Files\system\ado\msado26 .tlb
Microsoft Message Queue Object Library	D7D6E071-DCCD-11D0-AA4B-0060970DEBAE	1.0#0	C:\WINDOWS\system32\m qoa10.tlb
MessageQueueCo	B0C456C2-4DAE-11D4-	1.0#0	..\..\..\..\bin\VC\MessageQ

ComponentVC 1.0 Type Library	823C-005004E8D1A7		ueue\Windows2000\MessageQueueComponentVC.dll
Microsoft ActiveX Data Objects 2.1 Library	00000201-0000-0010-8000-00AA006D2EA4	2.1#0	C:\Program Files\Common Files\system\ado\msado21.tlb
ODIConverter 1.0 Type Library	12A83E98-ACDD-4FF9-87A8-F895DD747E07	1.0#0	..\ODIConverter\ReleaseU\nMinDependency\ODIConverter.exe
omStream 1.0 Type Library	E6BAA591-FE9F-11D4-82BE-005004E8D1A7	1.0#0	..\..\..\bin\VC\omStream.dll
Microsoft CDO for Windows 2000 Library	CD000000-8B95-11D1-82DB-00C04FB1625D	1.0#0	..\..\..\..\..\..\..\..\WINDOWS\System32\cdosys.dll
Windows Script Host Object Model (Ver 1.0)	F935DC20-1CF0-11D0-ADB9-00C04FD58A0B	1.0#0	..\..\..\..\..\..\..\..\WINDOWS\System32\wshom.ocx
omMutex 1.0 Type Library	B76FE60D-787D-4720-AADC-31202F934D72	1.0#0	..\..\..\bin\VC\omMutex.dll
Shared Property Manager Type Library	2A005C00-A5DE-11CF-9E66-00AA00A3F464	1.0#0	C:\WINDOWS\System32\COMSVCS.DLL
shdocvw.dll	EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B	1.1#0	
mscomct2.ocx	86CF1D34-0C5F-11D2-A9FC-0000F8754DA1	2.0#0	
comctl32.ocx	6B7E6392-850A-101B-AFC0-4210102A8DA7	1.3#0	

ACTIVEX CONTROLS

Control	Lib	Total
UpDown	MSComCtl2	1
WebBrowser	SHDocVwCtl	1
ProgressBar	ComctlLib	1

INTRINSIC CONTROLS

Control	Total
Form	3
Timer	2
CheckBox	2
ComboBox	3
TextBox	2
CommandButton	7
Label	5
Frame	1

WINDOWS API DECLARATIONS

Name	Lib
WinCoCreateGuid	OLE32.DLL
ShellExecute	shell32.dll
GetDesktopWindow	user32
DeviceCapabilities	winspool.drv
GetSystemDirectory	kernel32
pdfPrint	WPPdfSDK01.dll
EnumPrinters	winspool.drv
IstrlenA	kernel32
CopyMemory	kernel32
StringFromGUID2	ole32.dll
CallNextHookEx	user32
UnhookWindowsHookEx	user32
SetWindowsHookEx	user32
SetCapture	user32
GetClassName	user32
PostMessage	user32
Process32First	kernel32
Process32Next	kernel32
CloseHandle	Kernel32.dll
OpenProcess	Kernel32.dll
EnumProcesses	psapi.dll
GetModuleFileNameExA	psapi.dll
GetModuleFileName	kernel32
EnumProcessModules	psapi.dll
CreateToolhelp32Snapshot	kernel32
GetVersionExA	kernel32
GetSystemMenu	user32
RemoveMenu	user32
DrawMenuBar	user32
SetWindowLong	user32
CallWindowProc	user32
IsIconic	user32
FindWindow	user32
GetLongPathName	kernel32
Sleep	kernel32
SetWindowPos	user32
CreateProcess	kernel32
ShellExecuteEx	Shell32
WaitForSingleObject	kernel32
GetExitCodeProcess	kernel32
FindExecutable	Shell32
EnumWindows	user32
IsWindowVisible	user32
GetParent	user32
GetWindowThreadProcessId	user32
BringWindowToTop	user32
ShowWindow	user32
GetCurrentProcessId	kernel32

SetForegroundWindow	user32
GetTempPath	kernel32
UKCheckDetails	bw2uk
UKGetNextSubBranch	bw2uk
UKGetGeneral	bw2uk
UKGetBACS	bw2uk
UKGetCHAPS	bw2uk
UKGetCCCC	bw2uk
UKGetPrint	bw2uk
UKGetStatus	bw2uk
UKGetVersion	bw2uk
UKUpdateTables	bw2uk
UKSearch	bw2uk
RegCloseKey	advapi32.dll
RegCreateKeyEx	advapi32.dll
RegOpenKeyEx	advapi32.dll
RegQueryValueExString	advapi32.dll
RegQueryValueExLong	advapi32.dll
RegQueryValueExNULL	advapi32.dll
RegSetValueExString	advapi32.dll
RegSetValueExLong	advapi32.dll
SHDeleteKey	Shlwapi.dll
GetFileVersionInfo	version.dll
GetFileVersionInfoSize	version.dll
VerQueryValue	version.dll
Istrcpy	kernel32.dll
FindFirstFile	kernel32.dll
FindNextFile	kernel32.dll
FindClose	kernel32.dll
FileTimeToLocalFileTime	kernel32.dll
FileTimeToSystemTime	kernel32.dll
CreateFile	kernel32
GetLastError	kernel32
SHFileOperation	shell32.dll
SetFilePointer	kernel32
WriteFile	kernel32
FlushFileBuffers	kernel32
VSAWinCoCreateGuid	OLE32.DLL
CompleteAuthToken	secur32
DeleteSecurityContext	secur32
FreeCredentialsHandle	secur32
FreeContextBuffer	secur32
GetProcessHeap	kernel32
HeapAlloc	kernel32
HeapFree	kernel32
GetVersionEx	kernel32
QuerySecurityPackageInfo	secur32
InitializeSecurityContext	secur32
AcquireCredentialsHandle	secur32
AcceptSecurityContext	secur32
GetComputerNameEx	kernel32.dll

GBOpen	gbhltv32.dll
GBClose	gbhltv32.dll
GBGetAddress	gbhltv32.dll
GBGetAddressFromDPS	gbhltv32.dll
GBGetAddressFromAKey	gbhltv32.dll
GBGetAKeyFromDPS	gbhltv32.dll
GBGetError	gbhltv32.dll
GBPostcodeAddress	gbhltv32.dll
GBGetNext	gbhltv32.dll
GBGetPrevious	gbhltv32.dll
GBUpdatePassword	gbhltv32.dll
GBGetAbbrCounty	gbhltv32.dll
GBGetAddressKey	gbhltv32.dll
GBGetBuilding	gbhltv32.dll
GBGetCountry	gbhltv32.dll
GBGetCounty	gbhltv32.dll
GBGetCountyRequired	gbhltv32.dll
GBGetDefine	gbhltv32.dll
GBGetDepartment	gbhltv32.dll
GBGetDepLocality	gbhltv32.dll
GBGetDepThorofare	gbhltv32.dll
GBGetDHACode	gbhltv32.dll
GBGetDoubleLocality	gbhltv32.dll
GBGetDPS	gbhltv32.dll
GBGetEasting	gbhltv32.dll
GBGetEstGridRef	gbhltv32.dll
GBGetHLTVersion	gbhltv32.dll
GBGetLocality	gbhltv32.dll
GBGetLocalityDetails	gbhltv32.dll
GBGetLAAreaCode	gbhltv32.dll
GBGetLACode	gbhltv32.dll
GBGetLAWardCode	gbhltv32.dll
GBGetMailSort	gbhltv32.dll
GBGetMosaic	gbhltv32.dll
GBGetNHSCode	gbhltv32.dll
GBGetNHSRegion	gbhltv32.dll
GBGetNorthing	gbhltv32.dll
GBGetOldMailSort	gbhltv32.dll
GBGetOrganisation	gbhltv32.dll
GBGetPAFVersion	gbhltv32.dll
GBGetPOBox	gbhltv32.dll
GBGetPostcode	gbhltv32.dll
GBGetSecurityCode	gbhltv32.dll
GBGetSubBuilding	gbhltv32.dll
GBGetThorofare	gbhltv32.dll
GBGetThorofareDetails	gbhltv32.dll
GBGetToolsVersion	gbhltv32.dll
GBSetAddressCase	gbhltv32.dll
GBSetRangeIndicator	gbhltv32.dll
GBSetBuildingRange	gbhltv32.dll
GBSetDataFileType	gbhltv32.dll

GBSetDllLevel	gbhltv32.dll
GBSetInAddressFmt	gbhltv32.dll
GBSetInDelimiter	gbhltv32.dll
GBSetInFmtStyle	gbhltv32.dll
GBSetListSorted	gbhltv32.dll
GBSetMaxElements	gbhltv32.dll
GBSetNoStreetInd	gbhltv32.dll
GBSetOutDelimiter	gbhltv32.dll
GBSetOutFmtStyle	gbhltv32.dll
GBSetOutTerminator	gbhltv32.dll
GBSetOutAddressFmt	gbhltv32.dll
GBSetPostcodeFmt	gbhltv32.dll
GBSetReturnCounty	gbhltv32.dll
GBSetUnwantedElements	gbhltv32.dll
GBIMAPI	GBNRTI32.DLL
GetProfileString	kernel32
QASInitialise	QASRVEA.DLL
QASErrorMessage	QASRVEA.DLL
QASErrorLevel	QASRVEA.DLL
QASVersionInfo	QASRVEA.DLL
QASDataInfo	QASRVEA.DLL
QASSystemInfo	QASRVEA.DLL
QASUpdateKey	QASRVEA.DLL
QASUpdateCode	QASRVEA.DLL
QASLicenseInfo	QASRVEA.DLL
QASAuthorise	QASRVEA.DLL
QAServer_Startup	QASRVEA.DLL
QAServer_Config	QASRVEA.DLL
QAServer_Shutdown	QASRVEA.DLL
QAServer_Request	QASRVEA.DLL
QAServer_Search	QASRVEA.DLL
QAServer_FreeResponse	QASRVEA.DLL
QAServer_ResponseSize	QASRVEA.DLL
QAServer_Response	QASRVEA.DLL