Recycling Oterro

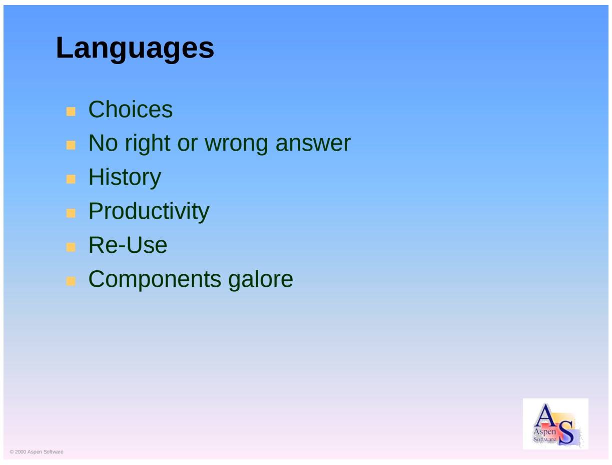
Neil Smith & Suzy Smith
Aspen Software

Summary

As we know, R:BASE 2000 provides a superb environment for the rapid production of database applications. Oterro gives us a way to benefit from the power of the R:BASE Engine when using Microsoft Visual Basic as our main application development tool. Using Visual Basic gives us access to incredible control and diversity in the production of an application. However, this does come at a price – compared to R:BASE there is much more work for the developer to do.

"Recycling Oterro" shows how we can start to close this productivity gap. Focussing on techniques such as COM object production and use, coupled with table driven structures, we will see how you can gain great productivity through real software re-use. Using the power of Oterro, Visual Basic and Active Server Pages with WebClasses, you will see how building block utilities can be harnessed and recycled to speed production of both Windows and Web based applications

Languages



Languages

- Choices
- No right or wrong answer
- History
- Productivity
- Re-Use
- Components galore

© 2000 Aspen Software

As developers we face constant choices about the environments we should use to build solutions for our clients. Learning and moving to new environments can involve a considerable investment of both time and money. Over the years, R:BASE, with one of the most complete database application development environments on the market has served us very well – and indeed, it continues to do so.

Given the nature of the IT industry, I believe that we should have open minds and be willing to consider other ways of doing things. All of us can learn from new ideas. Tempered with our own experience and healthy pragmatism we can apply these ideas to our own ways of working to give even better results.

Using the power of the R:BASE database engine with something other than the mighty R> prompt might be one such idea! Today, if we are developing "outside" an R:BASE application and want to access an R:BASE database, we use RBTI's Oterro. We can use Oterro with programming languages such as C++, Delphi and Visual Basic. This paper will focus on the use of Visual Basic and present a number of techniques that work for us, Aspen Software. Hopefully, they will work for you too. Please bear in mind though, that they are not the only the way we can achieve these results – as we all know, with software development there is not always a right or wrong way to get a result – just choices.

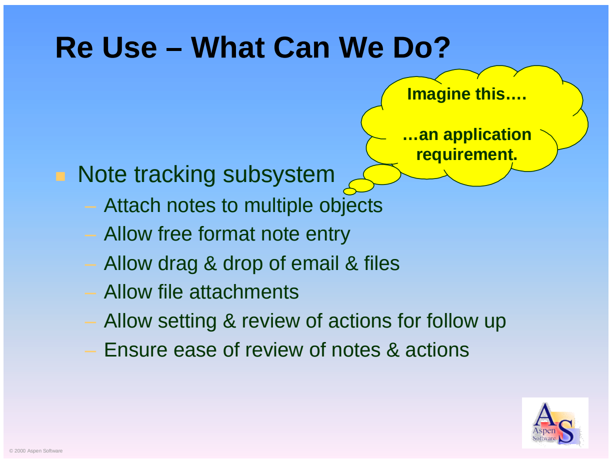
Microsoft released Visual Basic (VB) in 1990. Today, the product is at version 6.0 and version 7.0 (possibly to be called VB.net!) will be shipping next year. Visual Basic is probably the most widely used tool for the development of Windows hosted business solutions.

In 1994, Microrim released the "SQL Engine" a product that allowed us to access R:BASE 4.5 databases from the 16-bit version of VB. In 1995, around the same time as R:BASE for Windows 5.0, enhancements were made so that 32-bit VB could also access these databases. The SQL Engine was not without certain eccentricities, however, it still works today and there are legacy users who have active 4.5++ applications also being accessed by 32-bit Visual Basic applications.

Visual Basic also popularised the notion of using pre built components to help build your final application. Such components could be developed in house or they could be bought in "off the shelf". Indeed this has helped build a market for components and services that is expected to grow to $4.4 billion worldwide by 2002 (Source: Pricewaterhouse Coopers).

Visual Basic is positioned as a productive tool for creating fast business solutions for Windows and the Web. For example, using the component approach, if you had R:BASE data which you wanted to say: display in a Gantt chart, include in a mapped image, email to another user, or securely encrypt, then, while you could write the entire application yourself in VB, you would probably buy components to add the required functionality to your application.

Re Use – What Can We Do?



Re Use – What Can We Do?

Imagine this….
…an application requirement.

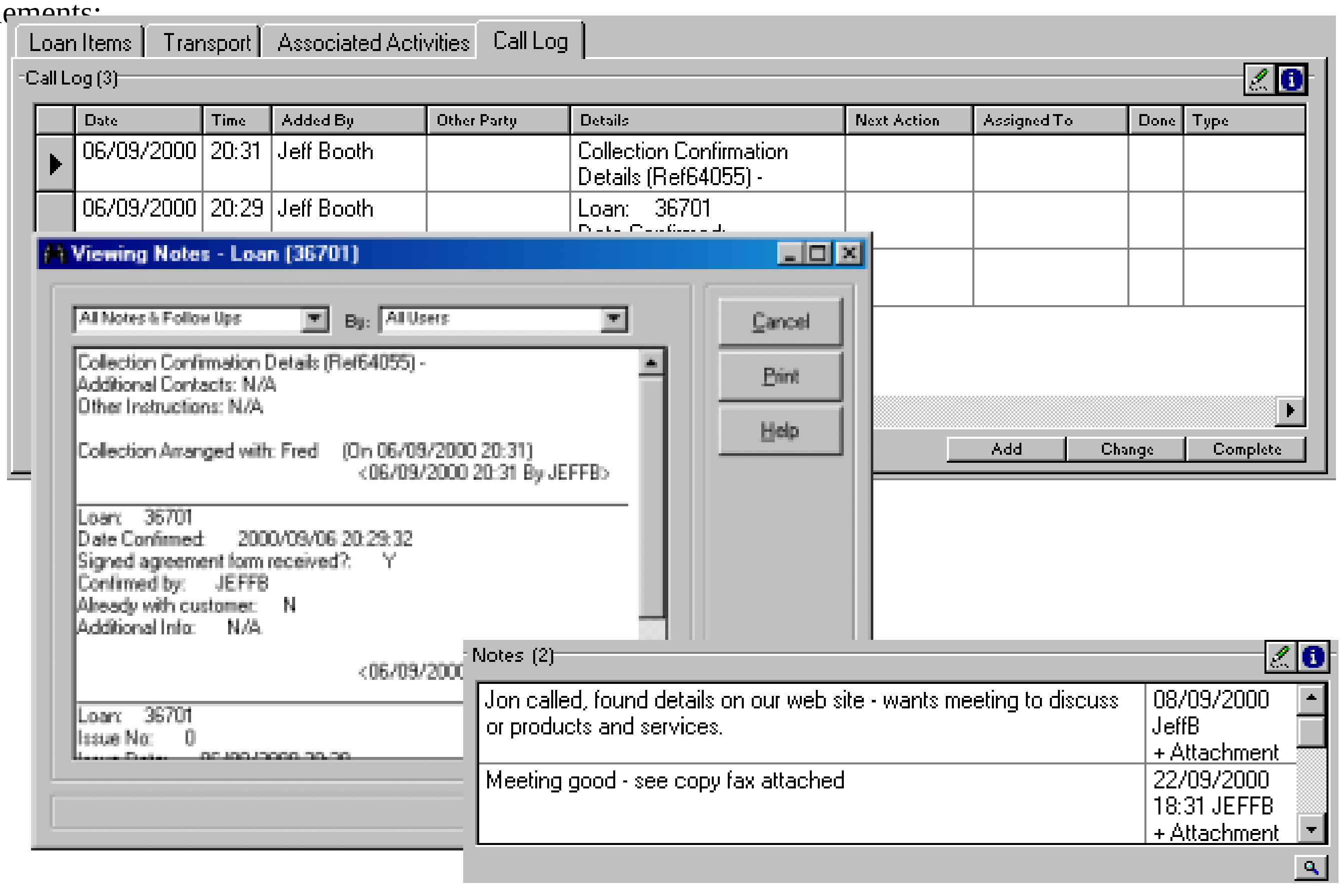
- Note tracking subsystem
  - Attach notes to multiple objects
  - Allow free format note entry
  - Allow drag & drop of email & files
  - Allow file attachments
  - Allow setting & review of actions for follow up
  - Ensure ease of review of notes & actions

© 2000 Aspen Software

Imagine a client requirement for a small part of an overall application. This is a Customer Relationship Based system and they want to log and track "notes" made by their representatives when communicating with their clients. Typically they want to:

- ➢ Make free format notes about any aspect of the customer's dealings with their organisation.
- ➢ Allow notes to be made from dragging and dropping emails.
- ➢ Allow files such as word documents and spreadsheets to be attached to the notes.
- ➢ Allow the setting of reminders to carry out actions at some future time.
- ➢ Allow an easy way of reviewing these actions and notes.

Overall, this is not an uncommon requirement, as developers it is likely we will all have done similar things many times for many customers. In this case, this part of the solution, which delivered all of the required functionality, comprised some of the following screens and elements:



Date	Time	Added By	Other Party	Details	Next Action	Assigned To	Done	Type
06/09/2000	20:31	Jeff Booth		Collection Confirmation Details (Ref64055) -				
06/09/2000	20:29	Jeff Booth		Loan: 36701 Date Confirmed:				

Viewing Notes - Loan (36701)

Collection Confirmation Details (Ref64055) -
Additional Contacts: N/A
Other Instructions: N/A

Collection Arranged with: Fred (On 06/09/2000 20:31)
<06/09/2000 20:31 By JEFFB>

Loan: 36701
Date Confirmed: 2000/09/06 20:29:32
Signed agreement form received?: Y
Confirmed by: JEFFB
Already with customer: N
Additional Info: N/A

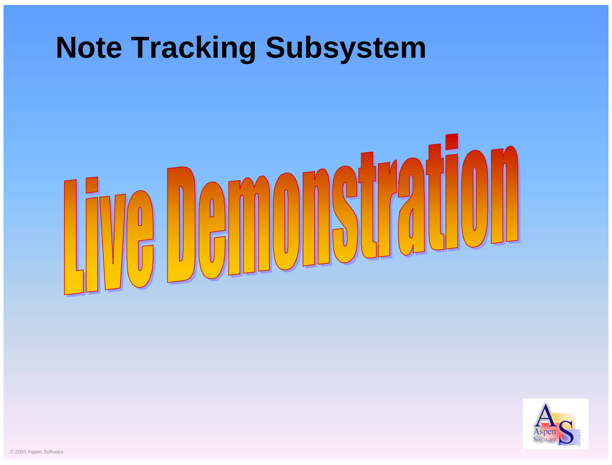
<06/09/2000

Loan: 36701
Issue No: 0

Notes (2)

Jon called, found details on our web site - wants meeting to discuss or products and services.	08/09/2000 JeffB + Attachment
Meeting good - see copy fax attached	22/09/2000 18:31 JEFFB + Attachment

Note Tracking Subsystem


Note Tracking Subsystem
Live Demonstration

As you will see, the real benefit using this approach occurs when we want to put similar facilities within another application.

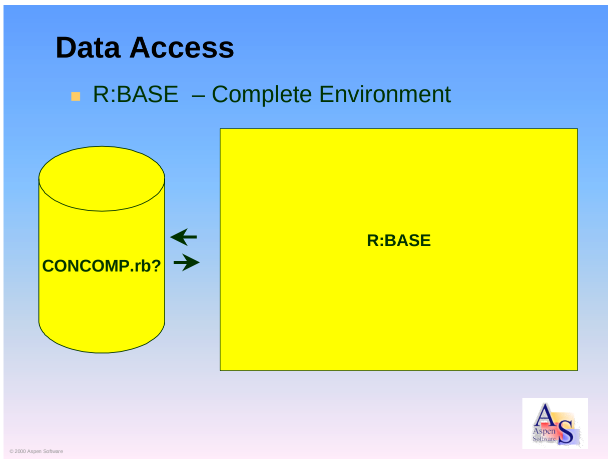
The functionality just illustrated was implemented within a component file (in this case an ActiveX control). This allows us to deploy a thoroughly tested and debugged application element that interacts with our main application in a well-defined way.

Core Technologies

In order to get this to work, and allow it to encompass web applications, we have to address three core technological areas:

- **Data Access** – how do we get the data from our R:BASE database into Visual Basic and back into R:BASE again?
- **Component Integration** – how do we actually write and deploy program components with the functionality we want that can be used by others?
- **Web Integration** – how do we then also use the components so they can be viewed across the web by any browser?

Data Access



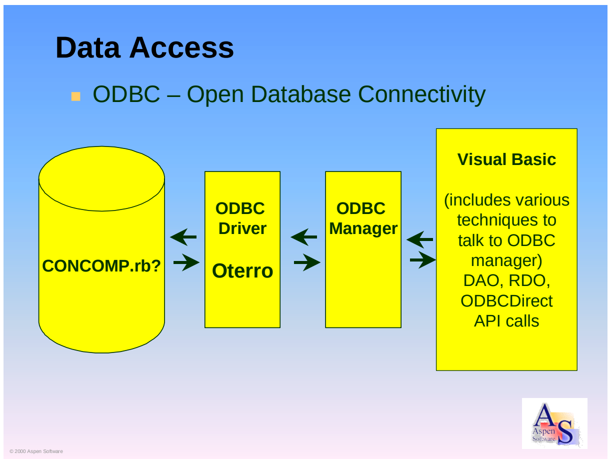
R:BASE shields us from the mechanics of getting the data from the database into our programming language.

```
SELECT varValue INTO vMyVar +
      FROM utConfig +
      WHERE varName = ’vLogDir’
```

Gives us immediate access to the value of the column varValue in our R:BASE application code. Another example would be:

```
CONNECT ConComp
UPDATE Product SET ListPrice = ListPrice * 1.1
DISCONNECT
```

ODBC



Outside of R:BASE, we need a “standard” way of getting to our data. Open Database Connectivity (ODBC) lets us achieve this.

ODBC is an application programming interface (API), with elements that typically look like:

```
Declare Function SQLConnect _
      Lib "ODBC32.DLL" _
      (ByVal hdbc&, ByVal szDSN$,_
       ByVal cbDSN%, ByVal szUID$, _
       ByVal cbUID%, _
       ByVal szAuthStr$,ByVal cbAuthStr%) As Integer
```

ODBC has been around a “long” time and it works. Developments started at a number of DBMS vendors in the late 1980’s. By 1994, when the R:BASE SQL Engine was released ODBC 2.0 was available. Today, the current version of ODBC is 3.52.

At its simplest, a Visual Basic database application will build strings of SQL commands, issue them against a database, fetch any returned data and store it within VB variables for subsequent processing.

To access a database from your Visual Basic program, the program will typically communicate with the ODBC manager “ODBC32.dll” (a Microsoft file). The ODBC Manager then communicates with the ODBC Driver (such as Oterro). Behind the scenes, Oterro comprises a number of files: Ot2k_32.dll, Ot2kins.dll, Ot2k1.msg, Ot2k2.msg, Ot2k3.msg and Oterro.cfg.

A further choice you have to make within Visual Basic is how you control the communication with the ODBC Manager. Your choices are:

- **Data Access Objects (DAO) –** this communicates via the Microsoft Jet (Access!) engine and presents a rich object model of your database (and slows everything down).
- **Remote Data Objects (RDO)** – this does not use Jet, and hence is quicker and provides a slightly different object model of your database.
- **ODBCDirect** – this uses the same object model as DAO but without using Jet.
- **ODBC API Calls** – forget about object models, this uses the raw calls of the ODBC Manager. It is the quickest way to get to your data – but is complex and less “friendly”.

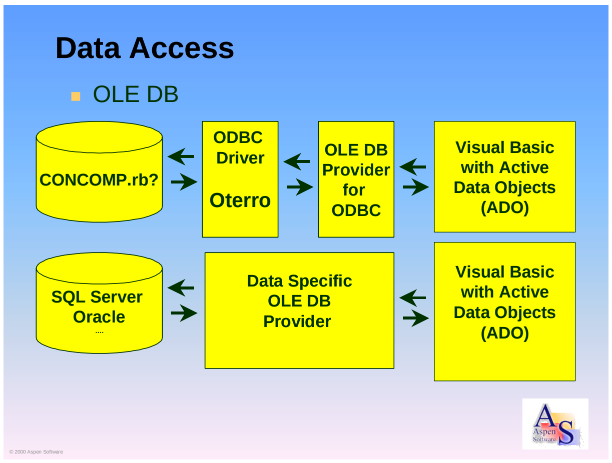
A raw example (with no error handling) of Visual Basic using ODBC API calls to connect, apply an update and disconnect is as follows:

```
'Allocate environment handle
retcode = SQLAllocEnv(DBEnv)

'Allocate connection handle
retcode = SQLAllocConnect(DBEnv, DBHandle)


'Connect
DBName = "CONCOMP"
retcode =  SQLDriverConnect(DBHandle, DBName, SQL_NTS, " ", 0, " ",
     0)
```

```
'Allocate statement handle
retcode = SQLAllocStmt(DBHandle, DBStmtHndl)

'Execute the command
SQLStatement = "UPDATE Product SET ListPrice = ListPrice * 1.1"
retcode = SQLExecDirect(DBStmtHndl, SQLStatement, SQL_NTS)

'Free the statement handle
retcode = SQLFreeStmt(DBStatementh, SQL_CLOSE)

'Close the database
retcode = SQLDisconnect(DBHandle)

'Free the connection handle
retcode = SQLFreeConnect(DBHandle)

'Free the environment handle
retcode = SQLFreeEnv(DBEnv)
```

The one key redeeming feature of this approach is that if you use compatible SQL commands, you can take the same piece of Visual Basic and run it against another vendor's database. In reality, unless you are very careful, engine specific SQL dialects creep in and need to be managed. For example `SELECT … FROM … WHERE LIMIT = 1` obviously works in R:BASE but does not in Microsoft SQL Server, in this case you need to use `SELECT TOP 1 … FROM ….` However, if you plan for these dialects, database independent applications can be developed.

OLE DB


Data Access
OLE DB
CONCOMP.rb?
ODBC Driver
Oterro
OLE DB Provider for ODBC
Visual Basic with Active Data Objects (ADO)
SQL Server
Oracle
….
Data Specific OLE DB Provider
Visual Basic with Active Data Objects (ADO)
© 2000 Aspen Software

OLE DB tries to make things a little simpler. It is positioned as Microsoft's strategic low-level interface to data across an organisation and is designed to build on the success of ODBC. This means you can also use OLE DB to access information that is stored outside a database - in other application files for example.

One key aspect of OLE DB, is that we communicate with "Providers" using COM (more about that later) and no longer need to use the API we illustrated above.

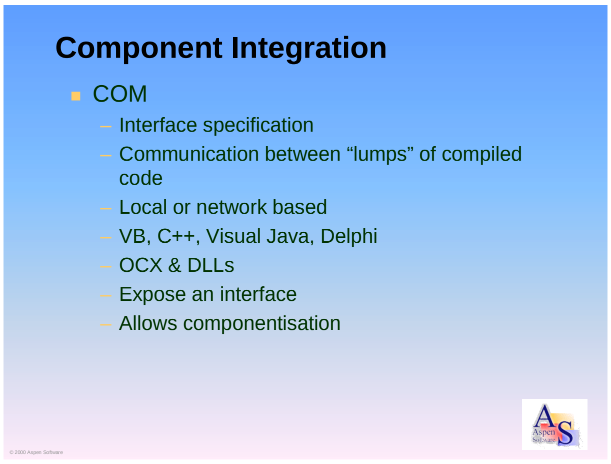
Today, if you are very lucky, a "native" OLE DB provider will exist that "talks" directly to the database (ORACLE and SQL Server being examples). For everyone else, a general purpose provider exists that will communicate with an ODBC driver.

Within Visual Basic, instead of the plethora of choices for ODBC, there is just the one approach Active Data Objects (ADO) that lets you control your OLE DB provider. Given this, the original example can now be reduced to:

```
'Set up ADO command objects
Dim MyCommand As Command
Set MyCommand = New Command
```

```
'Connect & issue commend
With MyCommand
   .Activeconnection = "Provider=MSDASQL.1;Data Source=ConComp"
   .CommandText = "UPDATE Product SET ListPrice = ListPrice *  1.1"
   .Execute
End With

'"Disconnect"
Set MyCommand = Nothing
```

COM



Component Integration

- COM
  - Interface specification
  - Communication between "lumps" of compiled code
  - Local or network based
  - VB, C++, Visual Java, Delphi
  - OCX & DLLs
  - Expose an interface
  - Allows componentisation

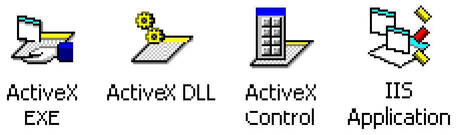
© 2000 Aspen Software

COM is the (Microsoft) framework that enables us to create and use components. It lets us define a "contract" with the users (other programs) of a component. The contract defines what information (Properties) can be read from or set in the component and what operations (Methods) can be carried out by the component.

It is a very much more structured and elaborate way of doing something like:

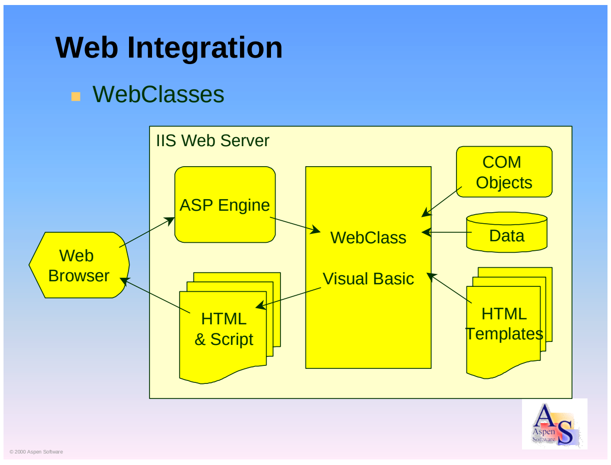
```
RUN MyComp IN COMPNENT.APX USING …
```

In the case of COM, components can be written in "any" language and can reside on the local machine or anywhere on a network.



Depending on the nature of the COM component it will either be implemented as an ActiveX EXE file, an ActiveX DLL file or an ActiveX Control (an OCX file). For Web Integration, IIS applications will be implemented as a WebClass DLL file.

Web Integration



The key thing to remember with Web enabling an application is that there are lots of ways to do it.

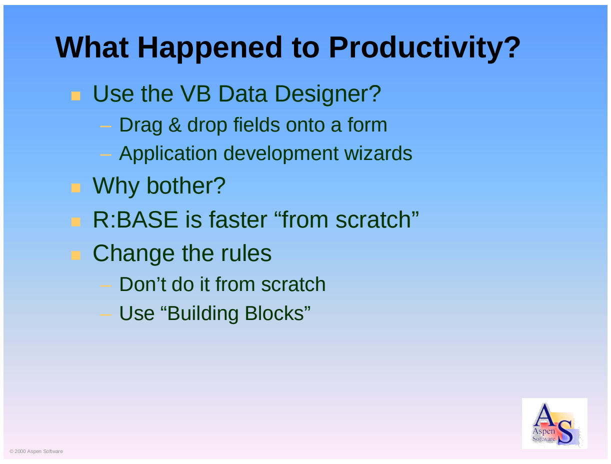
If we use a Microsoft IIS as our web server we can take advantage of Active Server Page (ASP) technology to harness and reuse our components.

On its own, we could write VB script in our ASP pages and use this to call COM objects that we have created using Visual Basic. This has some potential performance issues since an ASP page is stored in clear text on the server and interpreted at run time. Also, the Microsoft development environment for producing ASP pages (Visual InterDev) is not as fully featured as the Visual Basic development environment.

There is another way, which is to just use ASP as a vehicle to start something known as a WebClass. When a browser accesses a WebClass application, it requests an ASP page that simply starts a compiled WebClass DLL. As the WebClass DLL will have been written in Visual Basic, it can use other COM components to fetch data and apply business rules. It uses this intelligence to take HTML “template” files and customise them “on the fly” with data and application session specific information. This produces a final HTML file that is passed back to the browser.

The beauty of this approach is that our intelligence is in the VB WebClass (which is just another sort of COM object). Since it merges information into a template HTML file it also provides good separation between data management and the presentation of the page to the user. VB developers can produce WebClasses against raw HTML pages while the HTML authors design the final pages. Indeed, HTML templates can be changed after the WebClass has been compiled and deployed.

What Happened to Productivity?



What Happened to Productivity?

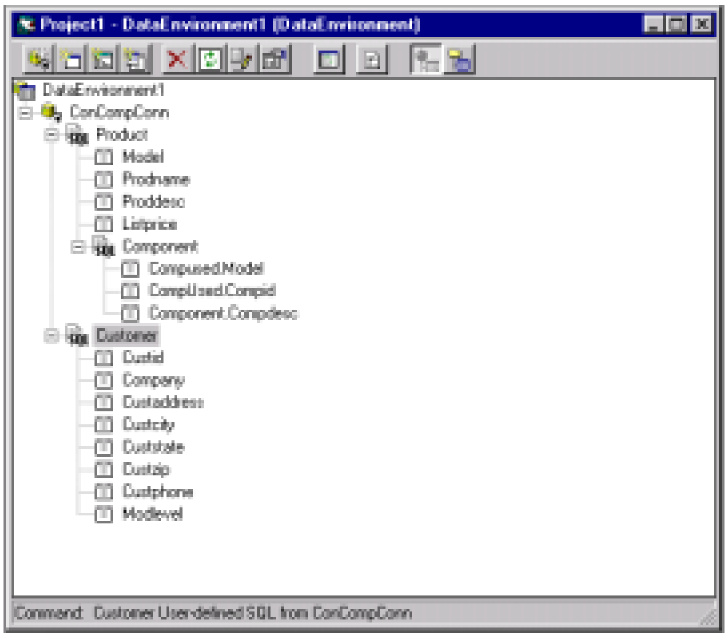
- Use the VB Data Designer?
  - Drag & drop fields onto a form
  - Application development wizards
- Why bother?
- R:BASE is faster “from scratch”
- Change the rules
  - Don’t do it from scratch
  - Use “Building Blocks”

© 2000 Aspen Software

All these technologies are very interesting, but consist of lots of layers. Don’t you just end up writing code that moves information backwards and forwards between layers? Is this really rapid application development!

As you will see Visual Basic comes with lots of goodies to address this. So working with Oterro you can drag and drop fields from the database onto forms and you can even use Application Wizards to create an initial VB application (which you subsequently edit).

To be blunt though, if you are going to work this way, you might as well work completely in an R:BASE environment. If you are going to build an application "from scratch"; R:BASE will be quicker and more productive.



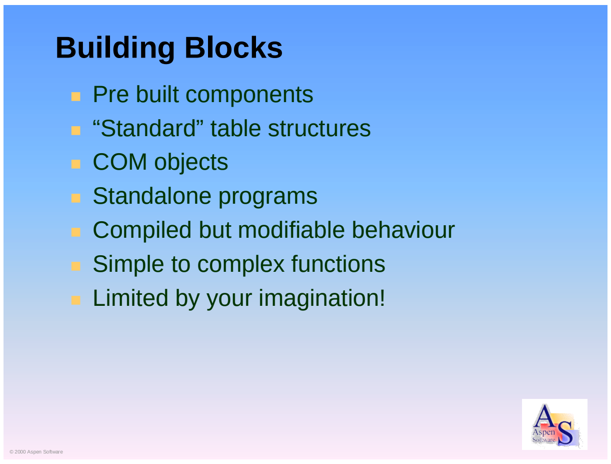
Project1 - DataEnvironment1 (DataEnvironment)

```
DataEnvironment1
⊟ ConCompConn
   ⊟ Product
      Model
      Prodname
      Proddesc
      Listprice
      ⊟ Component
         CompusedModel
         CompUsed.Compid
         Component.Compdesc
   ⊟ Customer
      Custid
      Company
      Custaddress
      Custcity
      Custstate
      Custzip
      Custphone
      Modlevel
```

Command: Customer User-defined SQL from ConCompConn

To get the real benefit of Oterro and Visual Basic, you will need to change the rules and use "Building Blocks". Instead of building an application from scratch, you are building it from tried and tested parts. This leaves you more time to focus on those bits of the application that are unique to the customer.

Building Blocks



Building Blocks

- Pre built components
- "Standard" table structures
- COM objects
- Standalone programs
- Compiled but modifiable behaviour
- Simple to complex functions
- Limited by your imagination!

So what exactly are these "Building Blocks"?

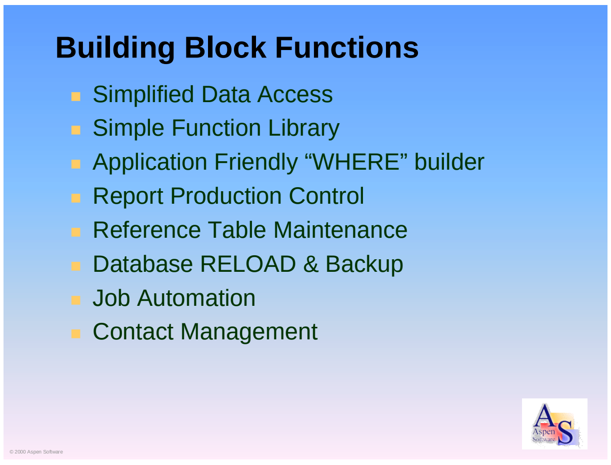
They are a combination of pre-built and tested COM objects (OCXs and DLLs), sets of known "standard" tables within our database and additional standalone programs.

An important general feature of most of the building blocks is that the properties and methods of the objects and the contents of the "standard" tables can be used to make the component seem like part of an overall dedicated application.

You can either build these components yourself or buy them in from a third party.

The facilities the building blocks provide can range from a library of simple routines to a complete and complex subsystem. You really are just limited by your imagination.

Building Block Functions



Building Block Functions

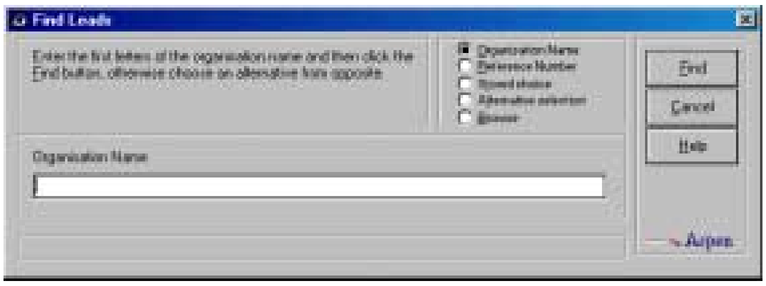
- Simplified Data Access
- Simple Function Library
- Application Friendly “WHERE” builder
- Report Production Control
- Reference Table Maintenance
- Database RELOAD & Backup
- Job Automation
- Contact Management

© 2000 Aspen Software

The aims of the building blocks are to provide functionality that we normally expect from a complete environment such as R:BASE or to hide complexities of the underlying technology we are using.

For example, we could implement a COM based library of ODBC database access routines. So the earlier example of ODBC code could be simplified to:

```
' Initialise the database engine
DBEngineInit()
'Connect
DbConnect()

'Execute the command
SQLStatement = "UPDATE Product SET ListPrice = ListPrice * 1.1"
DBSQLUpdate(DBHandle, DBStmtHndl, SQLStatement)

'Close the database
DbClose()
'Clear the DB Engine resource
DBEngineClose
```



Enter the first letters of the organisation name and then click the Find button, otherwise choose an alternative from opposite.

[✓ Organisation Name] Reference Number Stored choice Alternative selection Browse

Organisation Name

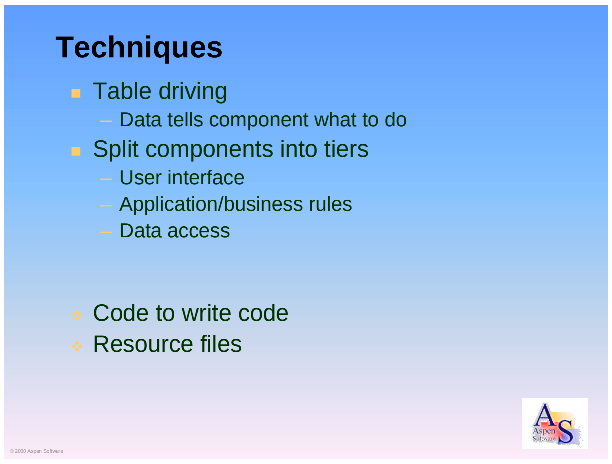
Other building blocks could provide the rich functionality of an application specific “WHERE” builder, give users an easy way of requesting and printing reports or provide a simple way for them to control configuration settings in their application.

A highly complex building block could be the implementation of a complete contact management subsystem.

Examples of standalone program building blocks would be a program that carries out a robust database RELOAD process or a program that copies and compresses database files as part of a backup process.

One element of R:BASE that we take for granted in an application that is also a prime candidate for a building block, is a way of maintaining the table of reference data that usually accompanies an application – the sort of table that you might want to maintain with `EDIT ALL FROM …`

Techniques



Techniques

- Table driving
  - Data tells component what to do
- Split components into tiers
  - User interface
  - Application/business rules
  - Data access
- Code to write code
- Resource files

© 2000 Aspen Software

This brings us to some key techniques to help us control the behaviour of our building blocks and to maximise their reusability.

Perhaps one of the most important techniques (not just applicable to Visual Basic and Oterro) is the use of table driving. As mentioned earlier, we use the data in a table to control what the building block should do.

Also, to maximise reusability, it is vital to group appropriate facilities together in components. Typically, components would be split into tiers, one tier being specific to User Interfaces of a particular type, another providing the business and application intelligence (independently of the user interface) while a third could provide the low level data access routines.

Another less important, though useful technique, is having "code that writes code" which is then executed. So in R:BASE we could use something similar to:

```
OUTPUT vTempFile
WRITE 'SET VAR vlogDir TEXT = ''\\AS-Server-01\App\LogFiles'''
OUTPUT SCREEN
RUN .vTempFile
```

When run, this would declare and assign the string value to the variable vlogDir. Unfortunately, Visual Basic applications usually run as compiled executable programs and so do not provide this interpretive capability. You can however use code to write code with WebClasses when producing JavaScript to run in a browser. As we will see later, Visual Basic enables us to change the nature of a form at runtime. This means that the more adventurous developers could produce their own simple interpretive routine that could then be fed code generated at runtime by the Visual Basic program. This is an ideal way to produce simple questionnaire forms that vary according to information held within the database.

Before looking at table driving and tiered code in more detail, another useful technique available within Visual Basic is the ability to compartmentalise the storage of a program's strings and images within a "resource file" which then forms part of the EXE file or a COM component. This enables you to isolate client specific splash images and logos within a COM component. You can then have client specific COM components holding different images that alter the look of the overall application without the need to recompile.

Table Driving – Application Configuration

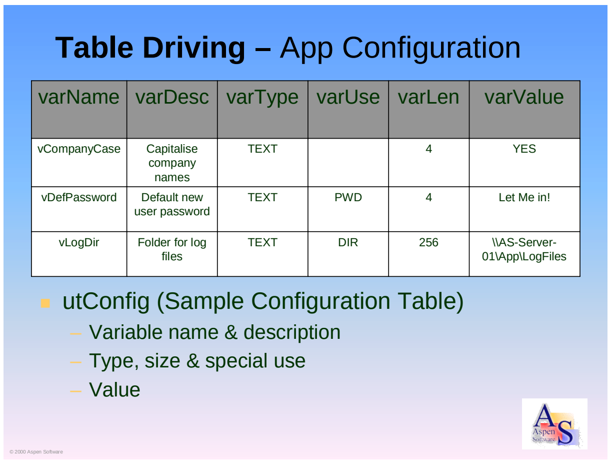


Table Driving – App Configuration

varName	varDesc	varType	varUse	varLen	varValue
vCompanyCase	Capitalise company names	TEXT		4	YES
vDefPassword	Default new user password	TEXT	PWD	4	Let Me in!
vLogDir	Folder for log files	TEXT	DIR	256	\\AS-Server-01\App\LogFiles

- utConfig (Sample Configuration Table)
  - Variable name & description
  - Type, size & special use
  - Value

© 2000 Aspen Software

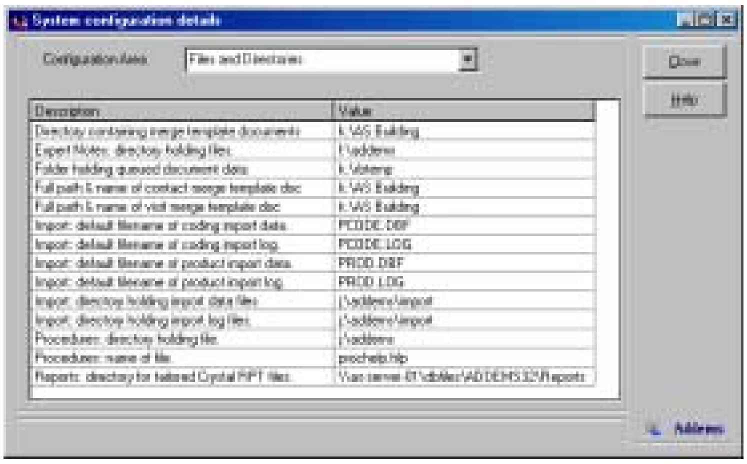
A simple example of table driving is the following structure used to hold configuration values for an application.

As you can see, the table holds columns for the name of our configuration variable, a description and details concerning its type size and use. A final column then holds the value to be assigned to the variable.

In an application that is using both R:BASE and Visual Basic to access the database, the R:BASE application can use either `SELECT … INTO` for individual values or OUTPUT to a file and use a selection to build a sequence of “SET VARS” which could then be run to create the variables.

The Visual Basic application would use a library routine from a component to access an individual configuration value. Eg:

```
vLogDir = GetConfigValue("vLogDir")
```



Description	Value
Directory containing merge template documents	k:\AS Building
Expert Notes: directory holding files	i:\address
Folder holding queued document data	k:\Visiting
Full path & name of contact merge template doc	k:\AS Building
Full path & name of visit merge template doc	k:\AS Building
Import: default filename of coding import data	PCODE.DBF
Import: default filename of coding import log	PCODE.LOG
Import: default filename of product import data	PROD.DBF
Import: default filename of product import log	PROD.LOG
Import: directory holding import data files	j:\address\import
Import: directory holding import log files	j:\address\import
Procedures: directory holding file	j:\address
Procedures: name of file	prochelp.hlp
Reports: directory for tailored Crystal RPT files	\\as-server-01\dbfiles\ADDEMS32\Reports

As illustrated, within the Visual Basic component, a form would also be available to let users maintain the configuration details for their system.

Table Driving – Reference Tables

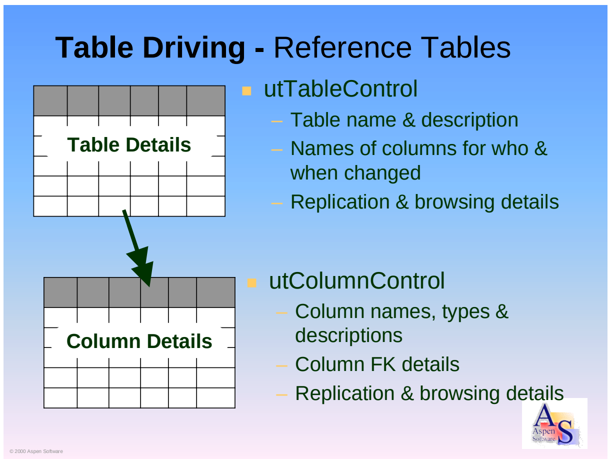


Table Driving - Reference Tables

- utTableControl
  - Table name & description
  - Names of columns for who & when changed
  - Replication & browsing details
- utColumnControl
  - Column names, types & descriptions
  - Column FK details
  - Replication & browsing details

© 2000 Aspen Software

As Visual Basic will let us change the characteristics of a form on the fly – you can build a common form and then alter it at runtime according to information retrieved from the database.

So if we hold details of the tables and the columns we want to maintain, we can then alter our form to display and update complete tables.

You could of course read this information from the SYSTABLE and SYSCOLUMNS structures in R:BASE or use ADO “properties” to find out these details. However, it may not be prudent to let a user change data in all the tables in the database. Holding your own structures does mean that you have more control over what the user can and cannot do. It also makes the component more portable allowing it to be easily implemented against other types of database engines.

Table Driving – Reference Tables

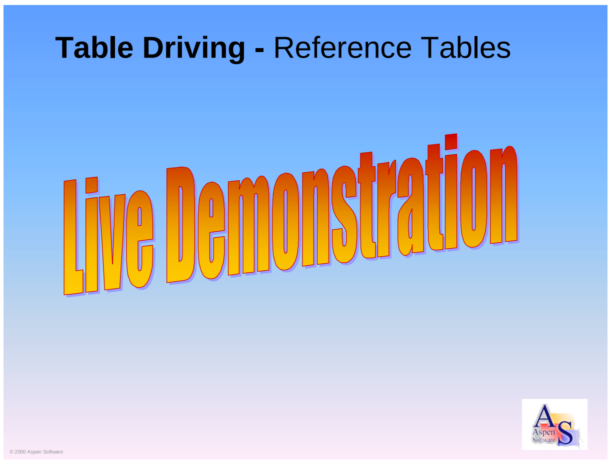
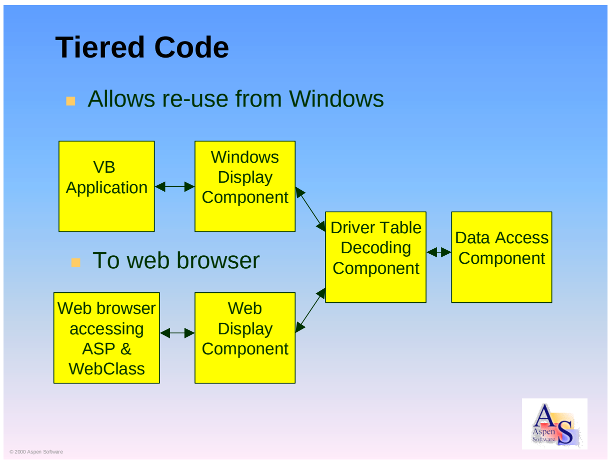


Table Driving - Reference Tables

Live Demonstration

As you will see, these structures allow us to control a powerful application maintenance tool. Tables and columns can then be easily added or removed from the maintenance screens.

Tiered Code



Tiered Code

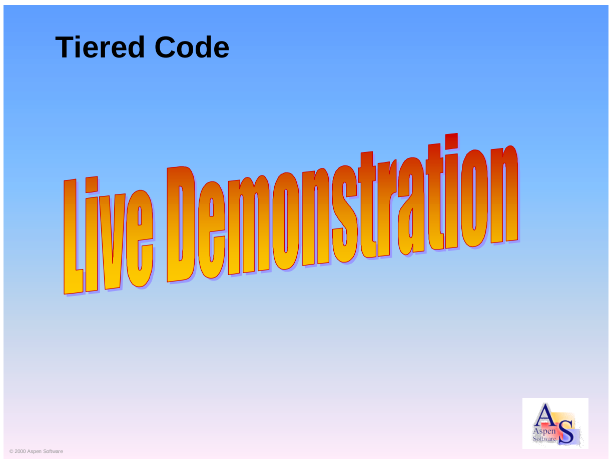
- Allows re-use from Windows
- To web browser

VB Application
Windows Display Component
Driver Table Decoding Component
Data Access Component
Web browser accessing ASP & WebClass
Web Display Component

If we structure our components into tiers, typically to split the User Interface, Business Application logic and Data Access methods we can then reuse our components in a number of environments.

The reference table component we have just seen splits the table decoding and data access components away from the code that deals with display in a Windows environment. When we come to implement similar functionality for a Web enabled application, we just need to produce a Web user interface element that uses the existing back tier components.

Tiered Code

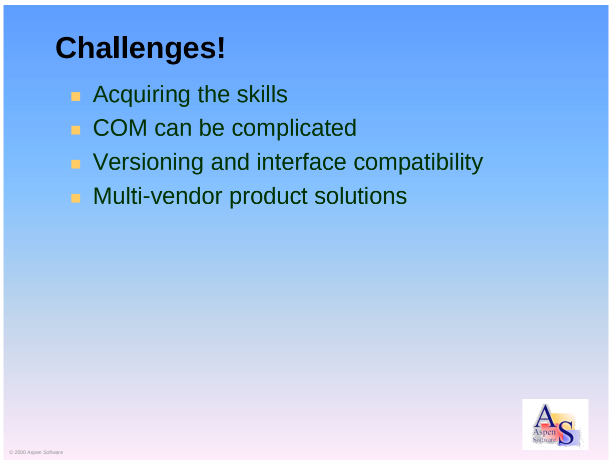


Tiered Code

Live Demonstration

So finally we see the equivalent Web functionality being delivered via a WebClass using the component files that have already been developed for use with our Windows application.

Challenges!


Challenges!

- Acquiring the skills
- COM can be complicated
- Versioning and interface compatibility
- Multi-vendor product solutions

© 2000 Aspen Software

As a "mere mortal application developer" I find it very exciting to be able to develop and deliver such incredible reusability with Oterro and Visual Basic.

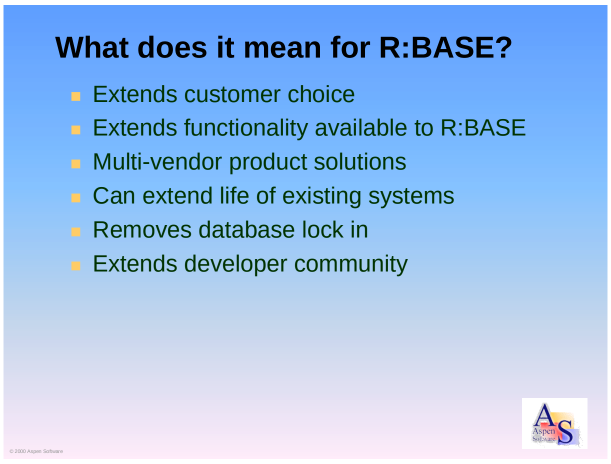
However, as with all things, it is not without its challenges. The biggest hurdle, if you have not already done so, is the investment needed to learn Visual Basic. Although if you are comfortable with complex programming tasks within R:BASE and are happy using an event driven approach such as that needed for R:BASE Entry Exit Procedures, then with the right resources, acquiring VB skills will not be daunting.

COM on the other hand, can be complicated (no pun intended) as we have an interface that allows components produced in different languages to communicate with each other on a local machine and across a network! Fortunately, Visual Basic does most of the hard work for you, thus simplifying the production of COM objects and, of course, there is a wealth of electronic and printed information readily available to help you.

When producing your COM objects, an area that needs careful consideration is the nature of the interface that your object provides – this is your contract with the outside world as to what your component will do. If, after releasing and deploying your component you then decide to "enhance" elements of the existing interface, add another parameter to an existing method for example, then you will have broken your contract! More critically, you will have broken the interface of your component. If you then replace the live "old" versions of your components with this new version, then applications will no longer be able to use the facilities of the component and will not work! (You will either need to recompile applications to use the version of the interface provided by this new component or, more usually, implement enhancements to the component as a completely new method.)

Finally, if you buy in components, which is usual, you are then involved in the delivery of a multi vendor product solution. When deploying, depending on your component provider this can either be a blessing or a challenge!

What does it mean for R:BASE?


What does it mean for R:BASE?

- Extends customer choice
- Extends functionality available to R:BASE
- Multi-vendor product solutions
- Can extend life of existing systems
- Removes database lock in
- Extends developer community

© 2000 Aspen Software

For any serious developer, we believe that the challenges are dwarfed by the benefits for both developers and customers when using these techniques.

Key to this approach is that you can give your customers almost whatever they want. If a pure R:BASE solution is not to their liking, then you can keep the power of R:BASE under the hood with Oterro and use Visual Basic. Having components available will speed the development of highly functional Windows and Web applications. The ability to rapidly add in specialised facilities via a third party component can give you the real advantages of a multi-vendor product solution – allowing you to choose the "best of breed" (as you do with the Oterro engine!) – and not settle for a compromised or inhibited application.

Where you have existing installations, the ability to use Oterro against databases that are already supporting R:BASE for Windows and R:BASE for DOS applications can open new territory and extend the life and reach of the system. For users who require specific facilities or interfaces that could not be provided by R:BASE alone, a dedicated Visual Basic and Oterro application can be developed to give those users exactly what they need, without detracting from the existing R:BASE applications.

Finally, for a customer it removes the fear of being locked in to one particular database vendor and having nowhere to go if their business exceeds the capabilities of the database engine. If your products and services are good then the removal of this "lock in" is not to be feared. From the R:BASE point of view I see this as an advantage, Oterro provides the engine of first choice, and customer's fears about it not being able to grow are allayed. Also, if you need to develop against another database engine then you can, using your existing components and giving you the capability to move the customer to Oterro when the opportunity arises.

Using Oterro with Visual Basic will enable us as R:BASE developers to extend our reach and also let us welcome Visual Basic developers who want to benefit from the power of Oterro.

What Next?

We have covered a lot of ground in this paper. Mastering the technologies and techniques discussed will enable you to re-use and capitalise on your skills. To find out more about the topics we have covered you may want try the following resources: (Please remember that this is not an exhaustive list and links may change over time.)

Oterro

- ✓ You can find out more from www.oterro2000.com/Products/Oterro.htm

Visual Basic and COM

- ✓ *Microsoft Visual Basic (Deluxe Learning Edition)* Michael Halveson, Microsoft Press; ISBN: 1572318732 is a complete learning package and includes a copy of Visual Basic 6 Learning Edition.
- ✓ msdn.microsoft.com/vbasic/ is the official starting point for Microsoft resources covering Visual Basic. The MSDN library CD subscription product is also an excellent resource with immediate access to over 15.GB of information (see msdn.microsoft.com/subscriptions/prodinfo/overview.asp)
- ✓ Carl & Gary's VB Home Page www.cgvb.com is an excellent source of "non biased" Visual Basic information and links – for new and expert users alike.
- ✓ *Doing Objects in Visual Basic 6* Deborah Kurata, Sams Publishing; ISBN: 1562765779. This introduces object oriented techniques and covers multi tier development. (We used the VB5 version of this book, which is no longer available. Some of the reader comments at Amazon may indicate better books to buy.)
- ✓ www.microsoft.com/com is the definitive site for information about COM
- ✓ *COM/DCOM Blue Book* Nathan Wallace, Coriolis; ISBN 1576104095. Not for the faint hearted, this is a highly recommended explanation of COM and Distributed COM.

OLE DB & ODBC

- ✓ *Oterro Manual.* In version 1.1 this covered all of the ODBC calls supported and how to use them. If you are going to be using ADO and OLE DB then you don't need to worry about ODBC calls.
- ✓ www.microsoft.com/data is the starting point for all information concerning Microsoft's data access technologies.

WebClasses & ASP

- ✓ *WebClasses from Scratch* Jesse Liberty, QUE; ISBN: 0789721260. A superb practical introduction to using all of the relevant technologies to produce and deploy an WebClass application.
- ✓ *VB Developer's Guide to ASP & IIS* AR Jones, Sybex International; ISBN: 0782125573. If you are doing WebClasses this is another "must have" leading on from WebClasses from Scratch and providing more detail concerning key areas.

Other Resources

- ✓ www.devx.com is a very useful site for answering "how do I" type questions, not just for Visual Basic but for other languages as well (including JavaScript).
- ✓ www.componentsource.com is the online source for finding and purchasing components, there is also an area dedicated to developers producing components.
- ✓ info@aspensoftware.co.uk if you have any questions about this subject.