



RAPID APPLICATION DEVELOPMENT TOOLS

PREPARED FOR MICROSOFT BY NSTL

JUNE 1998

RAPID APPLICATION DEVELOPMENT TOOLS 6/98

This report was prepared by NSTL, Inc. under contract for Microsoft, Inc. (Microsoft). NSTL does not guarantee the accuracy, adequacy or completeness of the services provided in connection with Microsoft's product. NSTL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO RESULTS TO BE OBTAINED BY ANY PERSON OR ENTITY FROM USE OF THE CONTENTS OF THIS REPORT. NSTL MAKES NO EXPRESS OR IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF ANY PRODUCT MENTIONED IN THIS REPORT.

EXECUTIVE SUMMARY

NSTL is the leading independent hardware and software testing organization in the microcomputer industry, dedicated to providing high quality services and test tools to the PC community. NSTL has extensive experience developing and conducting objective tests to assess new and existing products for compatibility, performance, usability, acceptance (bug) testing, and BIOS evaluation. NSTL's proficiency and thoroughness provide clients with a high-quality, cost-effective means to conduct this testing.

Microsoft, Inc. contracted with NSTL to compare leading Rapid Application Development (RAD) tools for Windows 95 and Windows NT. The products evaluated were: Visual Basic 6.0 from Microsoft Corp., PowerBuilder 6.0 from Sybase, Inc., and Delphi 4.0 from Inprise, Inc. (formerly Borland International).

USABILITY/PRODUCTIVITY RESULTS (SUMMARY)

Visual Basic was the easiest to use of the three products. Visual Basic has by far the best database development tools of the three products tested. Creating database forms and reports is as easy as dragging database fields from the database browser pane to your form or report. Visual Basic gives you complete control of your database within the development environment: you can modify and create tables, edit and debug stored procedures, and build SQL queries visually without losing your focus on the application.

Web sites are just as easy. To create your first web site you simply double click the app wizard (web site created) and click on "run". Beautiful! You can easily debug live web sites; clicking on debug pops you into Internet Explorer at the appropriate page. Visual Basic was the only program to specifically support COM with its debugger. Visual Basic provides the best set of online documentation -- complete with live hooks to its web site.

Delphi 4.0 improves productivity over previous versions with its new code browser and multi-project manager. Unfortunately the bugs and poor documentation still make for a steep learning curve. The database programming tools are the worst of the three. The Inprise SQL Explorer is a separate database browsing application with little integration with the development environment. The instructions for debugging with Microsoft Internet Information server are wrong, and the instructions for debugging with Transaction Server are not helpful. Configuring the Borland Desktop Engine is a trial and error affair with very little guidance from the documentation or administration tools.

PowerBuilder 6.0 is an improvement over previous designs, but it still has a very long way to go. Even PowerBuilder's much vaunted DataWindows are looking pretty tired compared to the new drag-drop database tools in Visual Basic. The heralded new debugger is superior to previous versions, but not up to Visual Basic or Delphi debugging standards.

PowerBuilder's web development model brings out the worst all the way around in the PowerBuilder development environment. The PowerBuilder web model is easy to learn, but tedious to use since

web sites can't be debugged live. On the positive side PowerBuilder offers an excellent integrated reporting solution and complete online documentation.

PERFORMANCE RESULTS (SUMMARY)

The performance matchup between Visual Basic and Delphi is pretty much a draw. Both provide good native code compilers. Visual Basic programming emphasizes ODBC and ADO database access, which is faster than Delphi's BDE. You can use "raw" ODBC and ADO in Delphi to achieve better performance, but at the cost of losing most of the visual support tools. Delphi's native ISAPI web model is faster than Visual Basic's COM based model, but both are very fast, and Visual Basic web sites scale better on multi-processor machines. Both compilers are incredibly fast; compilations are almost instantaneous. Delphi has changed very little in performance over the last two versions. Visual Basic 6.0 has made a significant performance boost (over 5.0) in data access with the new OLEDB drivers. In other respects its performance is similar to 5.0.

PowerBuilder is in a class all by itself when it comes to performance. It is slow in client/server, slower in three-tier development, and absolutely glacial in web applications. The ISAPI connection simply doesn't work, which compounds the performance problems. Version 6.0 offers no noticeable performance improvements over version 5.0.

FUNCTIONALITY RESULTS (SUMMARY)

Visual Basic shines in several important areas. Its debugger is superior in many ways. It can easily step through COM calls and into SQL Server stored procedures. Only Visual Basic allowed us to debug IIS applications without registry hacks. It likewise provides easy debugging for Microsoft Transaction Server (MTS) components. It has the most complete automation model for extending the development environment. Visual Basic is the only product of the three tested that supports native Alpha code generation, which is useful for creating very fast web servers.

Visual Basic carries the strongest bundle of "pack-ins". The Enterprise package bundles development versions of Source Safe, SQL Server 6.5, SNA Server, Microsoft Transaction Server, Microsoft Repository, Visual Modeler, and Microsoft Message Queuing server. Visual Basic provides interfaces to CICS, IMS, AS/400, and VSAM data sources in addition to the standard relational database interfaces.

Delphi provides a few features not available in Visual Basic or PowerBuilder. It has support for all the major web servers: ISAPI, NSAPI, CGI, and WIN-CGI. Delphi has integrated support for CORBA, Visual Basic and PowerBuilder do not. Delphi also offers advantages for advanced programmers with its built-in assembler, pointer support, and free threaded application support. Delphi has the weakest bundle of pack-ins. No source control package is provided, and the rarely deployed InterBase is provided as a SQL development platform. Delphi provides Midas and OLEEnterprise for 3 tier architecture support.

The PowerBuilder pack-ins (primarily ObjectCycle and SQL Anywhere) are not compelling. PowerBuilder supports native access to Lotus Notes databases in addition to traditional SQL data sources. PowerBuilder was the only product to offer support for executing code on Unix platforms, but either Visual Basic or Delphi can support Web applications on any platform that supports a browser. PowerBuilder offers meager support for web site development.

CONCLUSIONS AND RECOMMENDATIONS

Visual Basic provides the most powerful enterprise development package. Visual Basic provides excellent performance, is the easiest to learn and use, and provides an unmatched feature set. The new OLEDB drivers give it the fastest data access among the programs tested. The new database tools give it the strongest database programming tools of any development tool on the market. Visual Basic is the hands-down favorite for COM-related programming tasks. Creating ActiveX controls and Microsoft Transaction Server components is simplified by exceptional debugger support and the native COM object model. The new ADO support and Microsoft Transaction Server integration make it easy to use Visual Basic for three-tier applications. For web development Visual Basic's integration with ASP and IIS will appeal to the many developers already using that platform. For the 95% of Visual Basic developers that are programming database-centric applications, an upgrade to 6.0 is a no-brainer. The new version makes it easier to develop your program, and it makes the end product faster.

Delphi offers similar performance to Visual Basic, but lacks Visual Basic's compelling user interface and drag-drop database programming tools. We found the Borland Desktop Engine to be slower, less reliable, and less manageable than the OLEDB drivers used by Visual Basic. ActiveX and COM components in Delphi offer comparable performance to Visual Basic, but are more difficult to program. Delphi has some feature advantages for advanced programming, but most programmers would prefer to do this type of programming in C++. For Delphi 3 programmers, an upgrade would be mandated if you have urgent CORBA requirements. The other features like the code browser and minor Object Pascal enhancements are not that compelling.

PowerBuilder is a bad choice for developing anything for a Windows platform. It's tolerable for client/server development, but it is intolerable for web or three-tier development. The performance problems, bugs, and poor debugger (still bad, though much improved over previous versions) make it ill suited for this work. If you must work in PowerBuilder, the debugger alone is worth the upgrade.

RAD FOR CLIENT/SERVER

The bread and butter paradigm of corporate development is client/server programming: Client applications running on a Windows 95 connecting to a database server over the network. To evaluate each tool for client/server programming support we examined four essential capabilities:

1. Database Integration. How easily can you access the SQL (or non-SQL) data source from within the development environment? How easily can you create database queries and use them to build forms and reports?
2. Stored Procedures. How easy is it to manipulate stored procedures from within the development environment. How easy is it to create (and debug) code that uses stored procedures?
3. Report Generation. How easy is it to create WYSIWYG reports that can run as part of the finished application?

4. Extending the development environment. How easy is to create new ActiveX controls and COM components? How easy is it to add specific third-party COM/ActiveX components to the tool? How easy is it to build custom/proprietary components and tool add-in support utilities?

DATABASE INTEGRATION

The new Visual Database tools vault Visual Basic past Delphi and PowerBuilder in database accessibility. The database is available (non-modally) on your desktop at all times. The dockable "Data View" pane provides an Explorer-like tree view of the database that can be used to drag database objects into the programming environment. The pop-up menus in the data view provide the ability to create and modify database objects from within the development environment. Tables, views, stored procedures, and database diagrams are all conveniently within reach.

The new "Data Environment" provides a place to drop non-visual programming elements such as stored procedures and custom queries. The Data Environment provides a quick view into the database objects used by this particular program, where the Data View provides the view into the database's global state. The Data Environment also offers the capability to form database hierarchies for building drill-down reports. From the Data Environment you can drag database objects onto your visual application elements, including database forms and the spiffy new WYSIWYG report writer. Using a "right click drag" you can make instant choices about how the data is portrayed on the form: bound controls, data grid, or the new hierarchical grid.

PowerBuilder can do most of what Visual Basic can do within the development environment, but it is not nearly as well integrated or accessible. You must navigate the dialog boxes of PowerBuilder's exasperating modal interface instead of using direct manipulation.

Delphi offers relatively poor integration with popular SQL databases. The SQL Explorer included with Delphi has little integration with Delphi, and offers little value for database manipulation anyway. Most developers will prefer browsing tool provided by the database vendor (e.g. Enterprise Manager).

USING STORED PROCEDURES

Stored procedures are widely used to enforce security, business rules, and data integrity, and to improve performance. All of the programs offered at least some support for "wrapping" stored procedures with native programming objects.

Visual Basic provides the most advanced support for stored procedures, especially if you are using Microsoft's SQL Server. After one painless import operation we were able, to use all our stored procedures as if they were Visual Basic methods. Double clicking on a stored procedure within the Data View browser brings up the stored procedure in an editor/debugger. Stored procedures can be integrated into source code control together with the client application that calls them. Here is Visual Basic code for executing a stored procedure:

```
Visual Basic Stored Procedure Call
```

```
cn.is_valid_publisher "9999"
is_valid= cn.rs_is_valid_publisher(0)
```

 cn.rs_is_valid_publisher..Close

Delphi's support for stored procedures is more rudimentary. Stored procedures are linked one at a time into the project with arguments mapped to native Delphi types using a dialog box. The stored procedure can be executed, but not modified, within the Delphi development environment. Here is Delphi syntax for the same stored procedure:

 Delphi Stored Procedure Call

```
with is_valid_publisher_sp do begin
  ParamByName('@a').AsString := '9999';
  Open;
  Is_valid := Fields[0].AsInteger;
  Close;
end;
```

PowerBuilder has little visual support for stored procedures. They cannot be browsed or imported as a programming object. Instead, PowerBuilder offers special syntax for invoking stored procedures:

 PowerBuilder Stored Procedure Call

```
long id
declare is_valid_publisher_sp procedure for is_valid_publisher @id=:id;
execute is_valid_publisher_sp;
fetch is_valid_publisher_sp into :is_valid;
close is_valid_publisher_sp;
```

This approach has its advantages and disadvantages. It will be comfortable to programmers familiar with embedded SQL, but it will look foreign to programmers with visual programming/object oriented backgrounds.

FORM BUILDING

Visual Basic has the advantage over Delphi in this generation because of the new drag-drop database integration. Both can get you started with a basic database form wizard, but with Visual Basic it is easier to simply drag your query onto a form. Modifying forms is much easier within Visual Basic because the drag-drop operation condenses three operations into one: creating, naming, and linking the new control. Visual Basic also bundles a slightly richer set of controls, although neither product has clear superiority in this area. Visual Basic makes it a little easier to support custom data sources, and supports this in their form wizard. Delphi supports custom data sources, but making it integrate smoothly requires more work.

Visual Basic has adopted the COM/ActiveX object model as its native object model. No development environment on the market can rival its smooth integration of ActiveX controls. Delphi's designers have done a commendable job in making ActiveX components work in Delphi, but it still has a few rough edges compared to Visual Basic. The new class browser in Delphi 4 is an

important enhancement over 3.0, but it lacks the on-line help and method descriptions of the Visual Basic Object Browser. The pop-up completion editor helps are not as smooth or powerful as in the Visual Basic editor. Pascal doesn't offer argument passing by name, or call by name the way that Basic does, and this makes some controls tedious to use. We can't commend these features for their programming elegance, but useful objects exist that are helped by these features.

In version 6.0 Visual Basic also introduces a new form builder built around the capabilities of Internet Explorer. Called DHTML forms, they appear as web sites although the application runs locally (similar to the new HTML help system). These forms have the advantage of the rich formatting capabilities of Internet Explorer while using an event-processing model similar to traditional Visual Basic forms.

Delphi does offer some fine grain control that Visual Basic does not yet provide. Most of the native Delphi controls come with source code, and they get statically linked into the resulting form object. Aside from the comfort factor of being able to step through source code to find problems, the statically linked objects make the forms feel a bit "zippier". Reading the source is a great way to gain Delphi programming expertise. You can also use the provided source for the oldest form of code reuse: cut and paste.

PowerBuilder 6.0 has made some effort to spruce up the user interface, but has largely retained the same ineffective modal interface of versions past. PowerBuilder insists on showing you just one function at a time. Try to click on an object which is already open, and rather than going to that object PowerBuilder gives you an error message. The toolbars mutate in strange ways. You can't ignore them because many operations are available only from the toolbars.

PowerBuilder lags well behind Delphi and Visual Basic in ActiveX/COM integration. Only dynamic dispatch is supported. Errors that Visual Basic and Delphi catch safely at compile time cause run time failures in PowerBuilder. Performance suffers from the lack of static typing as well. PowerBuilder, despite its hefty price tag, carries the weakest bundle of controls.

REPORT GENERATION

Say what you want about the paperless office; management still wants reports. Report writers are a staple among client/server programming tools.

Visual Basic now sports the snazziest of the report writers, thanks to its clever new drag-drop database integration. The new report designer can be easily extended by adding new ActiveX components.

Despite this, the report generator remains a weak spot in Visual Basic's strong repertoire. Inexplicably the new report forms can only be used in an executable, and not in a DLL. The reports are not very useful to web site designers because Visual Basic web servers are always DLL's, and because HTML output can only be directed to a file. Cross tab reports still require the use of Crystal Reports.

The Delphi report writer shares many of the weaknesses of its Visual Basic counterpart, including poor web integration. Delphi's report writer is not as well documented, and it's not as easy to use. At least source code is available.

PowerBuilder still offers the most powerful reporting capabilities, and the only report writer that can be used directly from a web application. PowerBuilder supports cross tab reports, nested reports, and a few other attractive features. The user interface pales in comparison to the Visual Basic report generator, but at least it can generate more types of reports.

COMPONENT CREATION

RAD tools are best known as component consumers. Modern RAD tools offer the ability to craft new software components as well. Companies can enhance their client/server programming projects by using components customized to their activities.

Being able to create your own components is relatively new to client/server developers. Each of these development environments gained popularity by their ability to integrate components quickly into a finished product, but often times these components were difficult or impossible to create from within the tool itself. This is still largely true of PowerBuilder, which is largely ineffective for creating new controls and other components. Both Delphi and Visual Basic offer compelling capabilities for component creation.

Visual Basic makes it easy to create new controls. Controls are created with the same form designer that is used to create application forms. These controls are native ActiveX controls that can be immediately used in any ActiveX container. The main drawbacks are the hefty run time DLL's and threading limitations (apartment model only). Apart from the run time DLL's Visual Basic controls are usually smaller than their Delphi and PowerBuilder counterparts.

Delphi provides a control creation metaphor similar to Visual Basic. In addition Delphi 4 gives you the choice between small controls (dynamic linking) and zero runtimes (static linking). Zero runtime only works with non-database controls, or MIDAS based controls. All the database controls shipped with the product require the use of the Borland Database Engine (BDE). Not only must BDE be installed as a separate package, by license it must be installed with an Inprise certified installer (i.e. InstallShield variant). MIDAS allows you to install BDE on just the server, but it requires additional per server licensing fees. In this regard Delphi 4 has regressed from Delphi 3, which did allow you to ship controls based on server BDE configurations without additional licenses.

PowerBuilder components are only useful from other PowerBuilder applications. The limitations of using them in an enterprise environment with multiple languages outweigh any merit we can find. The components can't be statically typed and can't be debugged. You can only use the components on machines that have separately installed "PowerBuilder deployment packs". PowerBuilder has backed off its royalty per delivered application license, but the technology is still oriented towards a paid end user license.

RAD FOR ENTERPRISE

Each of the tools tested in some way claims to be appropriate for “enterprise” software development. While this can mean many different things, for this report we looked at three essential aspects of enterprise software development:

1. Ability to create new kinds of servers. For scalability and supportability it is often useful to create “business logic” components that are accessed through the enterprise WAN. These components may or may not run under transaction management software such as Microsoft Transaction Server.
2. Ability to support non-traditional data sources. A relatively small percentage of enterprise information resides in a relational database. An enterprise software solution must be able to integrate data from non-relational databases, flat files, and other non-SQL based data sources.
3. Ability to support team projects with dozens of programmers. Integration with source code control is a prerequisite, but other tools such as modeling tools and component repositories are fast gaining favor among large project teams.

CREATING NEW SERVERS

Another demanding new area for RAD tools is the design of “three-tier” applications. In three-tier logic there is a client, a “business logic” server, and of course the back-end database server.

Three-tier designs in Delphi and Visual Basic are based on DCOM, while PowerBuilder offers its own proprietary distributed object model. This choice reflects the Windows focus of Delphi and Visual Basic compared to the multiple operating system design goal of PowerSoft. At the time distributed objects were first made available in PowerBuilder, neither DCOM or CORBA was available on all the supported PowerBuilder platforms.

Creating three-tier logic in Visual Basic is disarmingly easy. Because COM is the native object model for Visual Basic, creating distributed objects is just as easy as creating local ones. Multiple COM components can be created in the same workspace, and then distributed once the code is working correctly. The debugger has many powerful features for multi-component testing. Stepping through a COM call can take you to another process and back again. Breakpoints can fire when your component is created by another application, even if you have not started that application under your debugger’s control. This is especially useful for testing components used by NT services. The thing that will occasionally haunt you is the single-threaded nature of the native development environment. You may have to occasionally resort to the bundled Visual Studio debugger. Unfortunately, this excellent multithreaded debugger is not yet integrated into the Visual Basic development environment.

A simple but important improvement in 6.0 is the addition of a second argument to CreateObject to allow the specification of a computer to create the object on (this required serious hack magic in Visual Basic 5.0). Another important upgrade in 6.0 is addition of ADO support. ADO allows result sets to be handed back from the server and then browsed on the client. New standalone ADO data sources (not backed by SQL databases) can be created as easily as stuffing data into an array.

Microsoft Transaction Server provides a great way to accelerate deployment of 3-tier applications. By using MTS you gain distributed transaction support and an array of security and performance management capabilities for little or no programming effort. Although MTS objects can be created using Delphi and PowerBuilder, Visual Basic offers the easiest path for developing MTS components. By setting one property in the object browser, your COM component becomes a full-fledged MTS component. Visual Basic's COM debugging support is a huge asset here: you can effortlessly step from client software through MTS into your server component as if no middle layer existed. By comparison, Delphi requires you to juggle two development environments (one for the client and one for the server) and PowerBuilder offers you no debugging in this environment.

It's a bit harder to create three-tier logic in Delphi. Native Delphi objects can't be simply handed around to out-of-process clients, although it is easy enough to create DCOM or CORBA objects for this purpose. A type library has to be built using a provided type library tool. Nothing too out of reach, but the documentation may have you scratching your head for a few minutes.

Delphi provides a similar feature to ADO in its MIDAS product. MIDAS offers a few more deployment options than ADO. Both can operate over DCOM or TCP connections (ADO over TCP requires the help of Remote Data Services (RDS)). In addition MIDAS technology supports deployment over CORBA and the OLEnterprise flavor of DCE. MIDAS is unique to Inprise products and is not likely to garner as much third-party support as the Microsoft-backed ADO model. Delphi's debugger can't step through an inter-process COM call like Visual Basic. This requires some additional setup to get debugging working correctly (especially with MTS).

MIDAS must be licensed at additional cost (unlike ADO). CORBA support also requires additional licensing fees beyond the cost of the compiler. It will be interesting to see if Inprise can make this stick in such a hotly contested market segment. It's not clear that either of these features are compelling enough for a developer to want to bother with additional expense and book keeping.

Delphi allows more flexibility in creating "free threaded" programs. Free threading can be a great asset in creating server objects. Hard core Visual Basic hackers can do all this too, but you do step out of the confines of your thread apartment at your own risk. The native Delphi debugger deftly handles multithreaded programs.

PowerBuilder gives you even less control over threading than Visual Basic, and the debugger is useless in a multi-component environment. The approach with PowerBuilder is to test the components in a single-user, non-distributed environment (always a good idea with any language) and then just hope that it works in the distributed, multi-component environment (not a good idea).

PowerBuilder's languid performance is crushing on server applications. On a fast client you might be able to squander some processor cycles running PowerBuilder scripts. You simply can't afford to use PowerBuilder on a server supporting hundreds of users simultaneously.

NON-RELATIONAL DATA SOURCES

The majority of enterprise data still exists outside of relational databases: flat files, legacy databases, or even new data stores such as directory servers or messaging servers. Each of the three products allows the programmer to bind controls to data sources other than relational databases.

Visual Basic and PowerBuilder approach the problem in a similar way: first populate a provided a generic record container object with data gleaned from the data source, then hand the populated data set back to the client. The essential part of this process is having a record container object that doesn't have to be connected to an SQL source.

Delphi's approach follows a more traditional OOP pattern: to create your own data source, you must implement the IProvider interface on your own custom object. This approach will be familiar to OOP programmers, but it can be more work and harder to understand initially. The payback for this approach is that you can generate more flexible data sources with higher performance. Visual Basic also supports this approach to creating OLEDB sources.

The other side of using non-relational data sources is getting the data into the server, in order to provide it to the client. Although people have pushed non-relational data through an ODBC spigot for years, ODBC has never quite fit this application. The SQL required for ODBC is a nuisance and a drag on performance if you are not dealing with an SQL data source. The new OLEDB/ADO model has a more flexible pattern for dealing with non-SQL data sources.

We found all the programs were able to deal with OLEDB data sources by writing code to access the ADO object model.

ADO in Basic

```

1. Dim rs As New ADODB.Recordset
2. Dim r As String
3. rs.Open "select * from publishers", "DSN=pubs;UID=sa;PWD=;DATABASE=pubs"
4. While Not rs.EOF
5.     r = r & rs(0)
6.     rs.MoveNext
7. Wend

```

Visual Basic was the only development environment to fully support ADO "out of the box". ADO sources are supported by all the database browsers and wizards.

ADO in Delphi

```

1 Var
2   rs: Recordset;
3   r: string;
4 Begin
5   rs:= CoRecordset.Create;
6   rs.Open('select * from publishers','DSN=pubs;UID=sa;PWD=;DATABASE=pubs'
7     ,adOpenForwardOnly,adLockReadOnly,adCmdText);
8   while not rs.EOF do
9   Begin

```

```

10    r := r + rs.Fields[0].Value ;
11    rs.MoveNext;
12    end;

```

Aside from cosmetic differences of the syntax, the Delphi code looks remarkably similar to the Visual Basic code. On line 6 you can see that we needed to provide arguments that are default arguments in Visual Basic and PowerBuilder, but now in Delphi 4 we could go in and tweak it to support default arguments. Default arguments are supported in Object Pascal starting with Delphi 4, but the type library import feature hasn't caught up yet. Considerable effort was required to bind ADO sources to Delphi controls. None of the visual tools in Delphi 4 support ADO.

ADO in PowerBuilder

```

1  OLEObject rs
2  string r
3  rs = CREATE OLEObject
4  rs.ConnectToNewObject("ADODB.Recordset.1")
5  rs.Open("select * from publishers", "DSN=pubs;UID=sa;PWD=;DATABASE=pubs")
6  do while Not rs.EOF
7      r = r + rs.Fields[0].Value
8      rs.MoveNext
9  Loop

```

PowerBuilder does not support static typing of COM interfaces. In our opinion that makes it unsuitable for anything but occasional ADO programming. While the syntax is commendably clean (and closest to the Visual Basic example), any syntax errors can only be caught at run time and not by the compiler. Performance suffers from the dynamic typing as well. PowerBuilder offers no editor "pop up" helps for navigating the syntax of foreign objects models.

TEAM PROGRAMMING

Each of the three products are targeted at multiple programmer, enterprise projects. At the very least that means source code control integration. A new twist being experimented with is the equivalent of source code control for pre-built software components: the component repository. And many OOP based projects are turning to visual modeling tools for class design. The most popular methodology for visual class design is based on the Unified Modeling Language (UML). UML supercedes the earlier Booch and OMT methodologies.

Visual Basic and PowerBuilder both provide a source code control system in the box, as well as the hooks to integrate them with other source code control systems. Visual Basic provides the widely used Source Safe product (also from Microsoft) while PowerBuilder provides the less recognized ObjectCycle (also from Sybase). Source Safe is the easier of the two to use and offers better integration with the project manager. ObjectCycle has the distinction of being the only source code control system that I know to be built on a relational database. This design makes ad-hoc reporting easier.

Delphi does not provide a source code system, but does provide the hooks to integrate with third party systems. Inprise marketing emphasizes Delphi's integration with the widely used PVCS.

Visual Basic offers the most complete repository capabilities and the only visual modeling tool of the three products. Visual Basic uses the Microsoft Repository, which encompasses multiple languages and object models. It also offers the Visual Component Manager, which offers a GUI interface to the repository for inserting and extracting components. The Microsoft Repository itself is hosted in a relational database. This design makes it easy to do ad hoc reports and extensions. The visual modeling tool is a fairly complete subset of Rational Rose for Visual Basic. Rational Rose is the most popular of the visual modeling tools on the market.

By comparison PowerBuilder and Delphi offer repositories for their own objects only. Delphi's is not based on a relational database, so reporting is not as easy. Neither product offers a visual modeling tool.

Visual Basic also provides the Application Performance Explorer. This tool lets you benchmark various configurations of two tier and three tier solutions using different databases and interfaces (e.g. ADO vs. ODBC). It's a useful idea to understand and benchmark these alternatives before beginning your programming. It's not the kind of tool you use every day, however.

RAD FOR WEB SITES

Most new enterprise programming projects involve the Web as an essential component. Although client/server remains an important part of the existing enterprise infrastructure, RAD tools are now being retargeted toward building Web-deployed applications. All three RAD tools tested for this report offer support for web deployed applications. All support approximately the same level of abstraction, similar to standard Web CGI development. Each tool provides an easy way to parse HTTP requests and a graphical interface for designing database queries that can be displayed using HTML. None of the tools provide a capable HTML editor or HTML form designer. External tools can be used; we were able to easily use InterDev, FrontPage, and of course, Notepad. Each product also allows GUI database forms and reports to run in browsers as Netscape Plugins or ActiveX controls.

Visual Basic was by far the easiest from box to deployed web site. No configuration changes were needed and the web startup wizard produced a working web site that could be subsequently modified. Debugging the application launches the web site and simultaneously opens it in Internet Explorer. Beautiful! InterDev users will be pleased at the level of integration between Active Server Pages (ASP) and Visual Basic. It's easy to tune your ASP site by rewriting heavily trafficked pages in Visual Basic. The completed application runs as a COM object under ASP, so a minimal completed application consists of a self-registering COM object, a manager COM object, the Visual Basic runtime DLLs, and a stub ASP page that Visual Basic generates automatically. Performance for the completed web site was excellent.

Visual Basic makes it trivial to track user sessions using cookies. Following the standard ASP model, the first time a user hits the web application, a session cookie is automatically generated. The web object can then refer to the Session system object to retrieve and store information about the current session. Sessions are timed out automatically as well. With either of the other applications, the web developer must write cookie tracking code if cookies are to be used. Using cookies with PowerBuilder is especially awkward.

There are a few flies in Visual Basic's ointment. We couldn't access the new report writer from our Web DLL. To do this we had to launch an ActiveX exe server and make out-of-process calls to it. Even in the external exe, we could only write the report to a file and not directly back to the browser. It's not worth the effort. The resulting web site was slower than a comparable C++/ISAPI based solution.

Delphi proved the biggest challenge to get working. With Delphi, nothing worked "out of the box." Here is a list of our problems and solutions (in case you are asked to develop a web site in Delphi):

1. Implement "Hello, world" web site. Unlike Visual Basic, the web site wizard does not create a working web site, but with book in hand we are running in under an hour.
2. Put site under load. It crashes.

3. Follow instructions for running IIS in a debugger. They don't work. Searched around in IIS documentation and found instructions for running IIS as a process. These work., and we have IIS running under Delphi's debugger.
4. Now the web site works under load up to 10 and crashes with a "too many connections error". Nothing in the printed documentation. Find reference in online help for property "Max Connections". Now "Hello, world" web site works with up to 100 threads.
5. Start on database stuff. Follow documentation, a little trial and error, get "BDE out of memory". Restart Delphi. Get "BDE Directory Locked" error. Start from scratch. This time, without going down the original blind alleys, it works.
6. Site under load now fails with "duplicate database name" error. Figure out how to create unique database names on startup. Wonder why database sessions can automatically get the unique names they need, but databases don't...
7. Site under load now fails with the message "BDE out of memory". Go to the BDE administrator help file. How much memory might BDE want? No clue. How much is it using? No clue. Trial and error with settings, a few hangs, a few error messages, and find a set of parameters that works. Hurray! It works under load.

Like Visual Basic, Delphi's reports are not readily accessible from web applications. Delphi does have a TQueryTableProducer component that allows simple database queries to be built visually and connected to your web site with a single line of code.

The main strength of Delphi compared to Visual Basic is the support for Web servers other than IIS. There is no supported integration with Active Server Pages, but it can still be accomplished with some effort. Delphi's deployment model is the simplest of the three; drop your executable in the scripts directory and you are installed.

Performance of the completed Delphi application is excellent in some tests, and unexpectedly poor in others. The Delphi runtime library doesn't scale very well on SMP machines. It has a difficult time taking advantage of more than one processor. A cursory look at the provided run-time source reveals a number of "critical sections" that may be slowing Delphi down.

The robustness of BDE in a server environment is potentially a problem. It would leak memory sporadically, causing the Delphi application to occasionally stop working (with every page returning "BDE out of memory"). One potential strategy would be to occasionally restart the web process (clearing out BDE), and another might be to avoid BDE. BDE also has the dubious design of sharing its limited memory among processes. This allows mission critical applications to be hosed by any application using BDE on the same machine. Even though BDE has this persistent state (it restarts only if all BDE applications are restarted), there is no tool provided to inquire as to its health and operating characteristics.

PowerBuilder proved too cumbersome for serious web site development. PowerBuilder is the only product that does not allow you to debug the application running from a browser. Instead each function must be tested as a standalone application, and then you must hope it works running from the web browser. Performance of the deployed applications is abysmal.

The robustness of the PB.Web solution is horrible. We couldn't run the ISAPI version for more than a few seconds without crashing the entire web server. Even when we resorted to the fallback CGI version we got sporadic protection faults in the PowerBuilder CGI stub (at least with CGI the web server stays running). The application throws another protection fault upon exiting. We cannot recommend PB.Web be considered seriously for even a moderately trafficked web site.

Unlike the other two products, a PowerBuilder web application must run in its own process. A PowerSoft-provided stub executable maps a web server-generated ISAPI, NSAPI, or CGI request into a PowerBuilder remote procedure call. This additional overhead reduces the server performance. Deployment consists of the PowerBuilder stub, a small executable to launch the application, a DLL for the actual application code, a DLL for the PowerBuilder web library, and the pbweb.ini file to glue it all together.

In testing the ISAPI and NSAPI versions using IIS and Netscape 3.51 respectively, we were never able to run very long without crashing. Sources within Sybase, and in companies using PowerBuilder 6.0, confirmed this same problem. No one could offer workarounds other than to use CGI. The performance of CGI was predictably dismal.

PowerBuilder was the only one of the three applications that could use visually designed reports effectively in web applications. All data windows can be output as HTML strings, making it trivial to use them in web sites. Power Builder was also the only application that can be used to deploy applications on Solaris, HP/UX, and AIX servers. Solaris, HP/UX, and AIX servers only support CGI and do not support all the web development features of PowerBuilder for Windows.

PERFORMANCE RESULTS

		Visual Basic	Delphi	PowerBuilder
Overall (geometric mean, bigger is better)				
Overall	Speed relative to PowerBuilder	9.3	9.0	1.0
SQL	Speed relative to PowerBuilder	1.8	1.2	1.0
String	Speed relative to PowerBuilder	2.7	1.9	1.0
Compiler	Speed relative to PowerBuilder	53	89	1.0
Web	Speed relative to PowerBuilder	18.2	19.1	1.0
SQL (milliseconds, smaller is better)				
	Call stored procedure	2.8	3.8	4.0
	Prepared query	2.8	4.2	8.9
	Complex Query: Compute median value	5.4	8.9	7.5

RAPID APPLICATION DEVELOPMENT TOOLS 6/98

value

Compiler (milliseconds, smaller is better)				
Compiler	COM call	1.8	.70	290
	Tak Function	5.9	5.4	210
	Sieve	17	8.2	490
	Permutations	160	120	7,800
String Tests (milliseconds, smaller is better)				
String	Format string	140	47	820
	Find all	0.4	11	21
	Find one (forward)	4.4	3.1	0.7
	Find one (backward)	3.4	3.3	1.0
	Replace all	100	100	1100
Web (pages/second, bigger is better)				
Web/SQL	Web Insert Record	58	110	2.5
	Web Query 23 Records	50	25	2.3
Web	Hello, world	120	230	11 ⁽¹⁾
	Hello, world x 1K	80	29	1.8 ⁽¹⁾
	Hello, program	120	120	9.5 ⁽¹⁾
	Shared State	110	230	9.5 ^(1,2)

Measurements to two significant digits (1%).

100 threads for ISAPI connections (Visual Basic and Delphi) and 10 threads for CGI (PowerBuilder).

Test system with two 200 MHz Pentium Pro processors

SQL Server 6.5

Windows NT 4.0, service pack 3, option pack (IIS 4.0)

All compiler optimizations enabled

¹ CGI application crashes

² Complete application crash

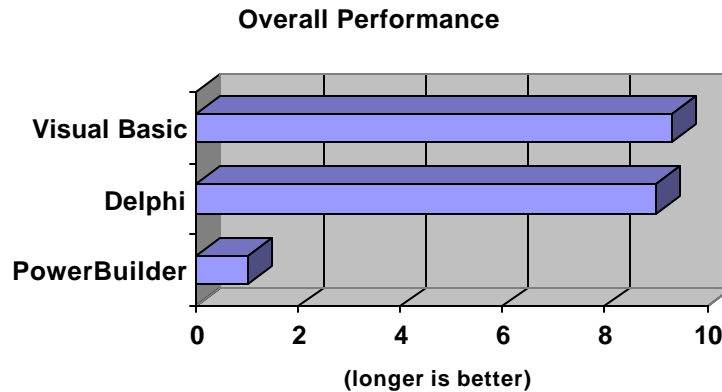
Delphi and Visual Basic both have native mode compilers that generate comparable code. Similar code compiled with Delphi runs a little faster than code compiled with Visual Basic, but the components delivered with Visual Basic (many coded in assembler and C++) are often faster than the components delivered with Delphi. Visual Basic's OLEDB based database access components are decidedly faster than the Delphi database access components based on its Borland Database Engine.

PowerBuilder generated the slowest programs. PowerBuilder was nine times slower than Visual Basic and Delphi for overall performance. It scored fifty times slower than Visual Basic and Delphi for raw code speed.

PowerBuilder ActiveX components are hobbled by a lack of early binding support. PowerBuilder was over 400 times slower than Delphi and nearly 200 times slower than Visual Basic using ActiveX components. We see no obvious way that early binding could be accomplished in PowerScript.

Forcing web sites through CGI makes PowerBuilder an order of magnitude slower than its competition. We had to reduce the number of concurrent threads to 10 to keep the CGI activity from swamping the server's capacity. Even with just ten concurrent "hits," our dual 200Mhz server was pegged at 100% CPU utilization for the entire test.

PowerBuilder's native mode compiler, much heralded in 5.0, is now downplayed in 6.0 (p-code is in vogue again). Although the code produced by the native compiler is slow, PowerBuilder p-code is slower yet (8-20x slower).



Description of Tests

Overall Speed		Take geometric mean of PowerBuilder time divided by product time.
SQL	Call stored procedure	Call a simple store procedure to check for the existence of a record
	Simple query	Same query as for stored procedure test, but without using the stored procedure
	Compute Median	Find the median quantity in a set of 21 records (Pubs.sales)
Compiler	COM call	10K calls to an in-process COM component.
	Tak(Integer)	If y is greater than or equal to x, Tak(x, y, z) is z. This is the non-recursive stopping condition. Otherwise, if y is less than x, Tak(x, y, z) is Tak(Tak(x-1, y, z), Tak(y-1, z, x), Tak(z-1, x, y))
	Sieve	Find prime numbers among first 100K integers. Repeat for 100 iterations.
	Permutations	Find all the permutations of the integers 0 to 8. This algorithm is useful for data mining and other optimization code. Taken from the C++ standard library.
Strings	Format String	Concatenate 1000 10 character strings converted from integers
	Find all	Find 100 instances of a two character string in a 10,000 byte string.
	Find forward	Find 10 byte string at the end of 10,000 character string.
	Find backward	Find 10 byte string at beginning of 10,000 character string starting from the back.
	Replace all	Replace 100 instances of a single character string with a two character string in a 10,000 byte string.
Web	Insert Record	Insert a single 40 byte record in response to a Web page hit. Pages per second using 100 threads.
	Query 23 Records	Create a HTML table from 23 records returned from SQL server (Pubs database, author). Pages per second using 100

Description of Tests

	threads.
Hello, world	A web page that generates a simple “Hello, world” dynamically. This shows the minimal overhead of a dynamic web page. Pages per second using 100 threads.
Hello, world x 1K	A large (16K) web page shows the transfer overhead of generated web pages. Pages per second using 100 threads.
Hello, program	Time to submit a form and return a generated response pages per second using 100 threads.
Shared State	Update an application counter to show how many application hits since the program has been up. Pages per second using 100 threads.

FUNCTIONALITY RESULTS

	Visual Basic	Delphi	PowerBuilder 6.0 Enterprise
COM Support			
Late Binding	Yes	Yes	Yes
Early Binding	Yes	Yes	No
Named parameters	Yes	Yes	Yes
In process servers	Yes	Yes	Yes
Out of process servers	Yes	Yes	No
Object Browser	Yes	Yes	No
Apartment threading	Yes	Yes	No
Free threading	No	Yes	No
Debugger			
Step into	Yes	Yes	Yes
Step over	Yes	Yes	Yes
Show call stack	Yes	Yes	Yes
Conditional Breakpoints	Yes	Yes	Yes
Watch variables	Yes	Yes	Yes
Show CPU registers	Yes (Visual Studio Debugger)	Yes	No
Debug machine code version	Yes (Visual Studio Debugger)	Yes	No
Show disassembly	Yes (Visual Studio Debugger)	No	No
Single Step through stored procedure	Yes	No	No

Multiple threads	No	Yes	No
Break on COM create object	Yes	In process only	No
Step through COM calls	Yes	No	No
Step through MTS wrapped COM calls	Yes	No	No
Remote debugger	No	Yes	No
Editor			
Extensible environment	Yes	Yes	No
Drag/Drop	Yes	Yes	Yes
Multiple Functions per window	Yes	Yes	No
syntax coloring	Yes	Yes	Yes
Function completion	Yes	Yes	No
Set breakpoints in editor	Yes	Yes	No
Language Issues			
Garbage Collect Objects	Yes	COM Objects	Yes
Class inheritance	No	Yes	Yes
Multiple abstract interfaces	Yes	Yes	No
Templates/Generics	No	No	No
Stack allocated objects	No	No	No
Method overloading	No	Yes	No
Operator overloading	No	No	No
Default parameters	Yes	Yes	No
Apartment Threading	Yes	Yes	No
Free Threading	No	Yes	No
Pointers to functions	Only for callbacks	Yes	No

Runtime type identification	Yes	Yes	Yes
Call by name	Yes	No	Yes
“Standard” calling convention functions	Yes	Yes	No
“C” calling convention functions	No	Yes	No
User defined union types	No	Yes	No
Pass user defined type by value	Yes	Yes	Yes
Pass user defined type by reference	Yes	Yes	Yes
Return user define type	Yes	Yes	Yes
Pass array by value	Yes	Yes	Yes
Pass array by reference	Yes	Yes	Yes
Return array	Yes	Yes	Yes
Exceptions	Yes	Yes	No
Inline assembly	No	Yes	No
Call Windows API	Yes	Yes	No callbacks
Use resource file for localization	Yes	Yes	No
Source for run time library is provided	No	Yes	No
Data Binding			
Bind controls to database queries	Yes	Yes	Yes
Bind controls to ADO/OLEDB objects	Yes	No	No
Bind controls to user created data sources	Yes	Yes	Yes
Bind to stored procedures	Yes	Yes	Yes
Create OLEDB data sources	Yes	Yes	No
Report Writing			
Output HTML	Yes	No, Queries can be formatted as	Yes

		HTML but not QuickReports	
Output Unicode	Yes	No	Beta
Cross tabulated reports	No	No	Yes
Compiler			
Can warn of unused variables	No	Yes	No
Can warn of using uninitialized variables	No	Yes	No
Can warn of unused assignment	No	Yes	No
Can warn of assignment to loop variable	No	Yes - Error	No
Can warn of functions not returning values.	No	Yes	Yes - Error
Optionally remove Pentium FDIV fix	Yes	Yes	No
Optionally remove array bounds checks	Yes	Yes	No
Create Windows API DLL's	No	Yes	No
Create standalone executables (no DLL's needed)	No	Yes	No
Targeted Platforms			
Windows 95/NT	Yes	Yes	Yes
Unix	Via HTML	Via HTML	Yes
NT/Alpha	Yes	No	No
Windows CE	Yes	No	No
Web Features			
Debug web server as service	Yes	No	No
Debug web server as application	No	Yes	No
Embed form in Browser	Yes	Yes	Yes
Debug form in browser	Yes	Yes	No

ISAPI	Yes (via ASP)	Yes	No
NSAPI	No	Yes	Yes
CGI	No	Yes	Yes
Connection pooling	Yes	Yes	Yes
Create downloadable controls	Yes	Yes	Requires separate install
GUI forms can be embedded in Web browser	Yes	Yes	Yes
Create local apps with web interface	Yes	Yes	No
Source provided for Web interface	No	Yes	Partial
Wizards/Code Generators			
Client/Server Wizard	Yes	Yes	Yes
3 Tier Wizard	Yes	Yes	Yes
Create Web application	Yes	Yes	Yes
NT Service	No	Yes	No
Wizard Wizard (generates wizards)	Yes	No	No
Property Page Wizard	Yes	Yes	No
Installation/Deployment Wizard	Yes	Yes	Yes
Web deployment wizard	Yes	Yes	Yes
Data form wizard	Yes	Yes	Yes
Stored procedure wrapper	Yes	Yes	No
Design/Analysis			
UML object modeler	Yes	No	No
Code Profiler	No	No	Yes
Design simulator	Yes	No	No

Component Profiler	Yes	No	No
Components			
Calendar	Yes	Yes	Yes
Tabs	Yes	Yes	Yes
Wizard	Yes	No	No
Grid	Yes	Yes	Yes
Tree	Yes	Yes	Yes
Control list (control repeater)	Yes	Yes	Yes
Hierarchical Grid	Yes	No	No
Chart	Yes	Yes	Yes
Modem/Telephony	Yes	No	No
FTP/HTTP/SMTP	Yes	Yes	HTTP only
TCP	Yes	Yes	No
Web Browser	Yes	Yes	No
MAPI	Yes	No	Yes
Lotus Notes	Through MAPI	Through SMTP/POP	Native API
Rich text	Yes	Yes	Yes
Progress	Yes	Yes	Yes
Servers Provided			
Database server	SQL Server	InterBase	SQL Anywhere
Source code control	Source Safe	None	Object Cycle
SNA	SNA server	None	None
Component repository	Yes	No	No

String Functions

Find substring	Yes	Yes	Yes
Reverse find substring	Yes	No	No
Search and Replace	Yes	No	No
Garbage collected strings	Yes	Yes	Yes

Data Management

Create/edit/view queries	Yes	Yes	Yes
Visually create/modify tables/views	Yes	Not through SQL	Yes
Edit stored procedures	Yes	No	No
Database diagramming	Yes	No	No
Integrate stored procedures with source control	SQL Server only	No	No

Component Creation

CORBA support	No	Yes	No
ActiveX controls	Yes	Yes	No
ActiveX documents	Yes	No	No
Netscape plugins	3 rd Party	Yes	Yes
Lightweight (windowless) controls	Yes	No	No

USABILITY RESULTS

	Visual Basic	Delphi	PowerBuilder
Debugger	Excellent	Good	Fair
Compiler speed (Debug)	Excellent	Excellent	Excellent
Compiler speed (Release)	Excellent	Excellent	Poor
Compiler warnings/errors	Good	Excellent	Poor
Editor	Excellent	Excellent	Fair
Language	Good	Good	Good
Installation/Deployment	Good	Excellent	Poor
Wizards	Good	Good	Good
Report Writer	Excellent	Fair	Good
Query Writer	Excellent	Fair	Fair
Database Designer	Excellent	Fair	Fair
Multithreaded support	Fair	Good	Poor
Code Browsing	Excellent	Good	Poor
SQL syntax	Good	Good	Excellent
COM interface	Excellent	Good	Fair
Win32 API	Good	Fair	Poor
Use cookies	Excellent	Good	Poor
Shared web state	Excellent	Good	Fair

Visual Basic is easier to learn than Delphi, and both Visual Basic and Delphi are significantly easier to learn and use than PowerBuilder. Each product took a similar amount of code to develop (see code in appendix A). Based on that, we would expect that experienced developers would achieve similar productivity levels with each product. Visual Basic and Delphi encourage better programming practice. Consequently, we would expect Visual Basic and Delphi developers to achieve better productivity in debugged lines of code per day than PowerBuilder developers.

The Visual Basic learning curve is easier than with Delphi. With the high turnover on programming projects these days it helps to have an environment that people can come up to speed quickly with. Delphi programmers can push that environment a little farther, but neither environment can challenge C++ based products for high complexity, high performance requirements.

Visual Basic has the best editor and debugger and proved to be the most reliable. Compilation is instant and the user interface is programmer friendly. The online documentation is excellent. Visual Basic's native object model is COM, so there is no better language for using or creating COM components like ADO or MTS components. Visual Basic is the easiest RAD tool for creating web sites.

We like the fact that Delphi can statically link executables. Visual Basic is designed to deliver applications that are dynamically linked together at runtime. This usually implies a setup program to distribute components and modify the registry. Run time problems can easily result from a mistake in the install, or worse, from a mistake in someone else's install. Delphi has the commendable characteristic of being able to generate useful standalone executables and DLL's that can be installed by simply copying to their final destination. Most commonly used components in Delphi are

statically linked into the executable. The most glaring exception is the Borland Database Engine, which has to be installed alongside most Delphi applications. By using the three-tier support you can push Delphi applications that don't need BDE to the clients, but the server will still need it.

Delphi has more powerful code analysis than either Visual Basic or PowerBuilder. You can probably find a number of errors in the following code, but Visual Basic can't find any. Delphi issues "hints" when it finds detectable programming errors. These hints are useful for finding these kinds of bugs before you even hit the start debugger key.

```
Private Function foo() As Long
  Dim x As Long
  Dim y As Long
  For x = 1 To 3
    x = 2
    Debug.Print x
  Next
  x = 5
End Function
```

PowerBuilder took us the longest time to develop the test code. The lack of debugger support for web pages forced us to develop additional test code. The debugger, while vastly improved over 5.0, is less integrated with the editing environment than either Delphi or Visual Basic.

The PowerBuilder development environment makes it difficult to browse through classes. It's difficult to see or print more than one function at the same time. Programmers faced with these obstacles invariably will write longer, less readable functions. The 6.0 environment is much improved over 5.0, but a long way from the competition.

CHOOSING A RAD TOOL

Each of these tools represents a significant advance over their previous versions. For maintaining existing projects, we can safely recommend that you upgrade to any of these tools from their previous versions. (A possible exception is PowerBuilder 6.0 for web projects; 5.0 is more stable under load.) For new projects you have a more difficult decision.

Programmers not experienced with any of these tools will find Visual Basic, the easiest to learn. Learning time is low productivity time, so Visual Basic improves your team's productivity by getting them doing productive work faster. What isn't in the box already (e.g., object modeler, component repository, and source code control) is readily available from third parties. If your team is already familiar with Visual Basic, then you have little reason to consider switching to either of the other tools. Visual Basic now has the database integration of PowerBuilder, and the native compilation speed of Delphi.

Programming teams familiar with PowerBuilder should seriously consider finding another tool. Visual Basic provides better database integration tools. Both Delphi and Visual Basic offer far superior compilers and debuggers. PowerBuilder's performance, always problematic, hits a brick wall

in implementing Web and multi-tier distributed software. Even experienced PowerBuilder developers will spend more time debugging (because of the debugger limitations) and tuning (because of performance problems) than their Delphi and Visual Basic counterparts. PowerBuilder programs cannot be easily retargeted or repartitioned to multi-tier environments because of PowerBuilder's poor characteristics for developing server code. PowerBuilder projects also have a higher than average chance of being scrubbed altogether because of insurmountable performance problems. This means lower productivity for generating useful enterprise applications.

Of the competing RAD environments, PowerBuilder developers will have the easiest time switching to Visual Basic. PowerScript is primarily a Basic variant (with a dash of C influence), and certainly far closer to Basic than to Java, C++, or Delphi's Pascal. The database tools in Visual Basic can be mastered in less than a day, so retraining costs will be minimized.

Experienced Delphi teams face the most difficult decision. Having hurdled the significant learning challenge, Delphi programmers can be just as productive as Visual Basic programmers can. Delphi's compiler is impressive: it generates fast code quickly. Delphi's COM integration is commendable, easily as good as any product other than Visual Basic (including Microsoft's other development languages).

Despite all its assets, Delphi remains just a bit outside the mainstream. It's hard to find good Delphi programmers. Training new ones is expensive, more expensive than with Visual Basic. Delphi packs the weakest collection of tools (no source code control, no modeler, weak repository) and third party tools are not readily available. Delphi's biggest drawback is the Borland Desktop Engine, which is not up to enterprise software standards of robustness and manageability. By comparison, Visual C++ and Inprise's own C++ Builder are no more difficult to learn than Delphi, and have the advantage of wide spread third party support. For new projects, programmers and managers using Delphi should seriously consider a switch to an easier environment (e.g., Visual Basic) or an equally difficult, but more powerful and standardized one (C++).