A

Seminar report

On

Windows DNA

Submitted in partial fulfillment of the requirement for the award of degree
Of MCA

SUBMITTED TO:

www.studymafia.org

SUBMITTED BY:

www.studymafia.org

Preface

I have made this report file on the topic Windows DNA; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to ……..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

Acknowledgement

I would like to thank respected Mr…….. and Mr. ……..for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

Secondly, I would like to thank my parents who patiently helped me as i went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs.

Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time.

CONTENTS

- INTRODUCTION
- MICROSOFT WINDOWS DISRTIBUTEDINTERNET APPLICATION
- TOP WINDOWS DNA PERFORMANCE MISTAKES and HOW TO PREVENT THOSE
- FEATURES AND ADVANTAGES OF WINDOWS DNA
- CONCLUSION
- REFERENCES

INTRODUCTION

For some time now, both small and large companies have been building robust applications for personal computers that continue to be ever more powerful and available at increasingly lower costs. While these applications are being used by millions of users each day, new forces are having a profound effect on the way software developers build applications today and the platform in which they develop and deploy their application.

The increased presence of Internet technologies is enabling global sharing of information—not only from small and large businesses, but individuals as well. The Internet has sparked a new creativity in many, resulting in many new businesses popping up overnight, running 24 hours a day, seven days a week. Competition and the increased pace of change are putting ever-increasing demands for an application platform that enables application developers to build and rapidly deploy highly adaptive applications in order to gain strategic advantage.

It is possible to think of these new Internet applications needing to handle literally millions of users—a scale difficult to imagine a just a few short years ago. As a result, applications need to deal with user volumes of this scale, reliable to operate 24 hours a day and flexible to meet changing business needs. The application platform that underlies these types of applications must also provide a coherent application model along with a set of infrastructure and prebuilt services for enabling development and management of these new applications.

Guiding Principles of Windows DNA

The Microsoft application platform consists of a multi tiered distributed application model called Windows DNA (Figure 1) and a comprehensive set of infrastructure and application services. Windows DNA unifies the best of the services available on personal computers, application servers, and mainframes today; the benefits inherent in client-server computing and the best of Internet technologies around a common, component-based application architecture.

The following principles guided Microsoft in developing the Windows DNA architecture:

- **Web computing without compromise.**

Organizations want to create solutions that fully exploit the global reach and "on demand" communication capabilities of the Internet, while empowering end users with the flexibility and control of today's PC applications. In short, they want to take advantage of the Internet without compromising their ability to exploit advances in PC technology.

- **Interoperability.**

Organizations want the new applications they build to work with their existing applications and to extend those applications with new functionality. They require solutions that adhere to open protocols and standards so that other vendor solutions can be integrated. They reject approaches that force them to rewrite the legions of applications still in active use today and the thousands still under development.

- **True integration.**

In order for organizations to successfully deploy truly scalable and manageable distributed applications, key capabilities such as security, management, transaction monitoring, component services, and directory services need to be developed, tested, and delivered as integral features of the underlying platform. In many other platforms, these critical services are provided as piecemeal, non-integrated offerings often from different vendors, which forces IT professionals to function as system integrators.

- **Lower cost of ownership.**

Organizations want to provide their customers with applications that are easier to deploy and manage, and easier to change and evolve over time. They require solutions that do not involve intensive effort and massive resources to deploy into a working environment, and that reduce their cost of ownership both on the desktop and server administration side.

- **Faster time to market.**

Organizations want to be able to achieve all of the above while meeting tight application delivery schedules, using mainstream development tools, and without need for massive re-education or a

"paradigm shift" in the way they build software. Expose services and functionality through the underlying "plumbing" to reduce the amount of code developers must write.

- **Reduced complexity.**

  Integrate key services directly into the operating system and expose them in a unified way through the components. Reduce the need for information technology (IT) professionals to function as system integrators so they can focus on solving the business problem.

- **Language, tool, and hardware independence .**

  Provide a language-neutral component model so developers can use task-appropriate tools. Build on the PC model of computing, wherein customers can deploy solutions on widely available hardware.

MICROSOFT WINDOWS DISTRIBUTED

INTERNET APPLICATIONS

Windows DNA: Building Windows Applications for the Internet Age

Windows DNA Technologies

The heart of Windows DNA is the integration of Web and client/server application development models through the Component Object Model (COM). Windows DNA services are exposed in a unified way through COM for applications to use. These services include component management, Dynamic HTML, Web browser and server, scripting, transactions, message queuing, security, directory, database and data access, systems management, and user interface.

Windows DNA fully embraces an open approach to Web computing. It builds on the many important standards efforts approved by bodies such as the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). Adhering to open protocols and published interfaces makes it easy to integrate other vendor solutions and provides broad interoperability with existing systems.

Because Windows DNA is based on COM and open Internet standards, developers can use any language or tool to create compatible applications. COM provides a modern, language-independent object model that provides a standard way for applications to interoperate at all tiers of the architecture. Through COM, developers can extend any part of the application via pluggable software components that can be written in C++, Visual Basic, Java, or other languages. Because of this open approach, Windows DNA supports a broad range of development tools today, including tools from Microsoft, Borland, Powersoft, and many other vendors.

Microsoft developed the Windows Distributed interNet Application Architecture (Windows DNA) as a way to fully integrate the Web with the *n*-tier model of development. Windows DNA

defines a framework for delivering solutions that meet the demanding requirements of corporate computing, the Internet, intranets, and global electronic commerce, while reducing overall development and deployment costs.

Windows DNA architecture employs standard Windows-based services to address the requirements of each tier in the multi tiered solution: user interface and navigation, business logic, and data storage. The services used in Windows DNA, which are integrated through the Component Object Model (COM), include:

- Dynamic HTML (DHTML)
- Active Server Pages (ASP)
- COM components
- Component Services
- Active Directory Services
- Windows® security services
- Microsoft® Message Queuing
- Microsoft Data Access Components

Microsoft built Windows DNA using open protocols and public interfaces, making it easy for organizations to integrate third-party products. In addition, by supporting industry-defined standards for Internet computing, Windows DNA will make it easier for developers to respond to technology changes. Some of the technologies recently added to the Windows DNA are outlined in the section given below, and are illustrated in the following diagram.

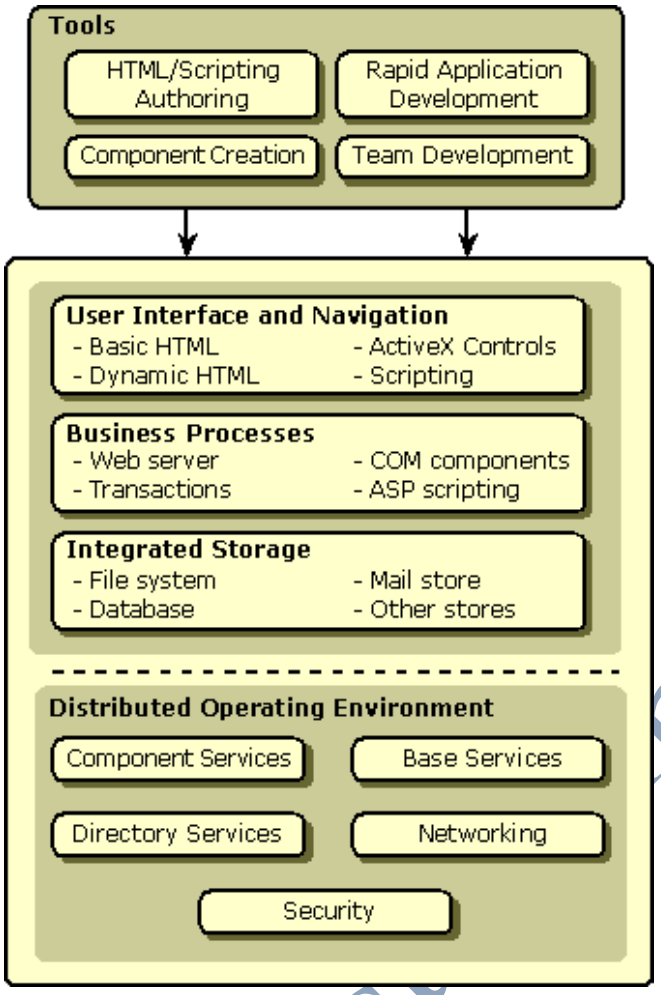

Figure 2. Technologies added to Windows DNA

Development Technologies

Microsoft Windows Distributed interNet Application (Windows DNA) Architecture is a dynamic set of technologies that you can use to build Web applications. Microsoft has added several key aspects to the architecture with Windows 2000.

This section contains:

- Component Services
- Dynamic HTML: Dynamic Hypertext Markup Language (DHTML).
- Windows Script Components
- XML: Extensible Markup Language (XML).

- Active Directory Service Interfaces

Component Services :

New with Windows 2000, Component Services provides a number of services that make component and Web application development easier. These services include:

Queued Components

Queued Components allow you to create components that can execute immediately if the client and server are connected. They provide an easy way to invoke and execute components asynchronously. In the event that the client and server are not connected, the component can hold execution until a connection is made. Queued Components assist the developer by using method calls similar to those calls used in component development, thus diminishing the need for an in-depth knowledge of marshaling.

Component Services Events

Component Services Events lets publishers and subscribers loosely connect to data sources so that these sources can be developed, deployed, and executed separately. The publisher does not need to know the number and location of the subscriber, and the subscriber uses an intermediate broker to find a publisher and manage the subscription to it. The event system simplifies component and Web application development by allowing both publisher and subscriber identities to be persistent: Publishers and subscribers identities can be manipulated without being known to each other.

Dynamic HTML :

Dynamic HTML (DHTML), which Microsoft introduced with Internet Explorer 4.0, allows you to create much richer HTML that responds to events on the client. By upgrading your HTML

pages to take advantage of DHTML, you will not only enhance the user experience, you will also build Web applications that use server resources more efficiently.

DHTML controls the appearance of HTML pages by setting properties in the document object model (DOM), a model which Microsoft has proposed to the World Wide Web Consortium (W3C) as a standard. DHTML exposes an event model that allows you to change DOM properties dynamically.

Windows Script Components :

Windows Script Components provide you with an easy way to create Component Object Model (COM) components using scripting languages such as Microsoft Visual Basic Scripting Edition (VBScript) and other languages compatible with the ECMA 262 language specification (such as Microsoft JScript 2.0 and JavaScript 1.1). You can use script components as COM components in applications such as Internet Information Services (IIS), Microsoft Windows Scripting Host (WSH), and any other application that can support COM components.

Script component technology is made up of the following:

- The script component run-time (Scrobj.dll).

- *Interface handlers*, which are components that extend the script component run-time. An interface handler is a compiled component (generally written in C++) that implements specific

COM interfaces. When you install the script component run-time, you will receive the Automation interface handler, which makes it possible to call your script component from an .asp file.

- Your script component file (a.sct file). In your script component, you specify which interface handler you want to use. Your script component also defines the methods that can be called from an .asp file to accomplish the intended functionality.

Script components are an excellent technology for developing prototypes of COM components. Script components, like any other COM component, can be registered with Component Services

if you intend for them to participate in transactions, or if you want to take advantage of the Component Services run-time environment. Because they are COM components, script components can access the ASP built-in objects.

Top Windows DNA Performance Mistakes and How to Prevent Them

Microsoft Windows DNA is Microsoft's platform for building a new generation of effective and versatile business applications for the Web. Through the COM+ programming model, Windows DNA incorporates a number of familiar technologies, including Microsoft Windows 2000, Microsoft Visual Studio, and Microsoft SQL ServerÃ‚Â™, allowing for the construction of a secure, stable—and scalable—business infrastructure that can readily integrate diverse systems and applications.

At the core of Windows DNA is the capability of building *n*-tier applications, which include one or more middle tiers between the client and the server. An important element of contemporary software architecture, *n*-tier applications provide clear advantages over typical client/server implementations, especially in the level of scalability they can provide. They are essential for the increasing levels of cross-platform interactivity required by today's, and tomorrow's, business Web sites.

Producing a good *n*-tier application often entails a series of judgments in planning and implementing the final product. When those decisions are poorly made, development teams can encounter time-consuming—and often difficult to solve—performance problems after the application has been installed and implemented. Fortunately, many of these problems can be anticipated and prevented. This article shows you how to find and eliminate them early in the development process. The mistakes that follow were identified by Microsoft Consulting Services (MCS) consultants worldwide. We've assembled some useful solutions, and while following them may not prevent all of the problems you'll encounter, you will significantly reduce performance degradation.

Misunderstanding the Relationship between Performance and Scalability

Performance and scalability are not the same, but neither are they at odds. For example, an application may process information at an incredibly fast rate as long as the number of users sending it information is less than 100. When that application reaches the point at which 10,000 users are simultaneously providing input, the performance may degrade substantially, because scalability wasn't high enough in the list of considerations during the development cycle. On the other hand, that same high-performance application may be partially rewritten in a subsequent iteration and have no problem handling 100,000 customers at one time. By then, however, a substantial number of customers may have migrated to a product someone else got right the first time.

Sometimes applications seek scalability in terms of number of concurrent users strictly through performance, with the idea being that the faster a server application runs, the more users can be supported on a single server. The problem with this approach is that increasing the number of simultaneous users may create a bottleneck that will actually reduce the level of performance as the load increases. One cause of this kind of behavior is caching state and data in the middle tier. By avoiding such caching in the design phase of the development process, countless hours of backtracking and rewriting code can be avoided. The ideal is to find a point of balance that provides acceptable performance in a scalable implementation of a particular application. Finding this point always involves trade-offs.

Let's look at some of the basic concepts involved in scalability. *Throughput* refers to the amount of work (number of transactions) an application can perform in a measured period of time and is often calculated in transactions per second (tps). *Scalability* refers to the amount of change in linear throughput that occurs when resources are either increased or decreased. It is what allows an application to support anywhere from a handful to thousands of users, by simply adding or subtracting resources as necessary to "scale" the application. Finally, *transaction time* refers to the amount of time needed to acquire the necessary resources, plus the amount of time the transaction takes actually using these resources.

The point to note here is that scalability increases as throughput increases; that is, the higher the throughput growth per resource, the greater the scalability. Clearly, application developers must concentrate their efforts on increasing throughput growth if they are to increase scalability. Of course, the obvious question then becomes, how exactly does one go about increasing throughput? The answer to that question may sound reasonably simple: Reduce the overall growth of transaction times. But just how easy might that be?

Transaction times can be extended by a variety of factors. Acquiring the necessary resources can be slowed by such factors as network latency, disk access speed, database locking scheme, and resource contention. Added to that are elements that can affect resource usage time, such as network latency, user input, and sheer volume of work. Windows DNA application developers should concentrate on keeping resource acquisition and resource usage times as low as possible. Frank Redmond lists the following ways to manage some of these factors:

- Avoid involving user interaction as part of a transaction.
- Avoid network interaction as part of a transaction.
- Acquire resources late and release them early.
- Make more resources available. Otherwise, use MTS to pool resources that are in short supply or are expensive to create.
- Use MTS to share resources between users because it is usually more expensive to create a new resource than to reuse an existing one.

Eliminating the confusion that exists about the relationship of performance and scalability, in this context, primarily means remembering that running a high-performance application is not the only consideration for gaining an acceptable level of performance in a Windows DNA application. It must be scalable so that the largest number of simultaneous users can be logged on without compromising throughput to an unacceptable level.

Mistakes in the Middle Tier

The next three performance mistakes relate directly to middle tiers in the Windows DNA application. A middle tier in an *n*-tier application is necessarily complex because of the role it

plays in the overall application. The specific tasks it performs can be separated into three general categories that are essential to Windows DNA applications.

The first task involves receiving input from the presentation tier. This input can be done programmatically or may come directly from a user. It may include information about (or a request for) almost anything. Second, a middle tier is responsible for interacting with the data services to perform the business operations that the application was designed to automate. For example, this might include sorting and combining information from different mailing lists to target a specific audience that was never previously considered to be a cohesive group. Finally, a middle tier returns processed information to the presentation tier so it can be used however the program or user sees fit. Within these three areas, performance can degrade significantly when developers use programming practices that are either little understood or mistakenly embraced as the "right" thing to do. These performance-compromising mistakes are explained more fully in the following sections.

FEATURES AND ADVANTAGES OF WINDOWS DNA

- DNA helps to design and build multi-tier client/server applications. It provides a structured approach to creating applications whose components are clearly separated into distinct functional groups, with common communication protocols linking these groups. This provides the benefits of faster and less error-prone design and development, and interchangeability of components.

- DNA provides client transparency. The front-end (or client) is independent of the back end of the application, i.e. it needs no knowledge of this, irrespective of what, the back end of the application does, or how it does it. As long as it follows the DNA protocol and processing guidelines, it can be almost anything—from a standard Web browser to a specially developed application written in almost any programming language.

- DNA applications provide full transactional processing support. In applications of any real level of complexity, multiple operations are performed at different levels of the application, and at different times. To guarantee integrity of the results, there needs to be control over each set of operations as a whole, as well as monitoring of every individual step. DNA, and the associated software plumbing components, can accomplish this almost transparently and seamlessly.

- DNA can be used to create applications that are fault tolerant. As no network can ever be 100% guaranteed to give continuous and fast performance. A distributed application needs to be able to cope with network delays and software failures, while protecting data integrity and providing high availability and reliability.

    - DNA is ideal for distributed applications. Once an application becomes distributed, i.e. divided into separate parts linked by a network, the problem of communication between the parts arises. Earlier it was necessary to create custom formats and techniques for passing information between each part of the application, leading to longer design and implementation periods, an increased number of bugs, and poor interoperability between

different applications. By standardizing the communication protocols and interfaces, development speed and application reliability is boosted.

The DNA methodology covers many existing technologies to help design and implement robust, distributed applications. It visualizes this whole application as a series of tiers, with the client at the top and the data store at the bottom. The core of DNA is the use of business objects in a middle tier of the application.

Also, in DNA, business objects are implemented as software components. These components can be accessed by the client interface application or by another component, and can themselves call on other components, data stores, etc. Componentization of business rules brings many benefits, such as easier maintenance, encapsulation of the rules, protection of intellectual copyright, etc.

Hence, DNA is an approach to design that can speed up overall development time, while creating more reliable and fault tolerant applications that are easily distributable over a whole variety of networks.

To run these applications, Windows DNA relies on a rich set of integrated services supplied by the Windows platform. These services are infrastructure technologies that would be required for any scalable, distributed application -- for instance, transaction processing, security, directory services and systems management.
By providing a stable base of common services, Windows DNA relieves developers from the burden of creating their own infrastructure and allows them to focus instead on delivering business solutions. Developers save time, reduce costs, get their applications to market more quickly and equip companies for responding proactively to changing business conditions. These benefits are especially compelling in today's competitive business climate.

Several more good reasons why companies should base their applications on Windows DNA. Because the architecture is built on open protocols and industry standards, solutions from other vendors integrate easily into the environment. This helps ensure interoperability with mission-critical business applications, such as corporate databases and enterprise resource planning systems. An open approach also facilitates compatibility with existing computing systems, which

means that companies can continue to take advantage of their legacy systems as opposed to replacing them.

The benefits of distributed computing applications that embrace Internet technologies are many. For individuals, it means the freedom to communicate or access information at any time and from any place. For businesses, it means making more informed decisions, better understanding their customers, and responding quickly as their business needs evolve. For software developers, the challenge has been how to build these solutions. Windows DNA offers them a cohesive and proven application architecture for distributed and Internet-based computing solutions.

CONCLUSION

The Windows DNA architecture and the Windows NT platform offer many distinct advantages to customers and their ISV partners. Its key benefits include:

- Providing a comprehensive and integrated platform for distributed applications, freeing developers from the burden of building the required infrastructure or assembling it using a piecemeal approach.
- Easy interoperability with existing enterprise applications and legacy systems to extend current investments.
- Making it faster and easier to build distributed applications by providing a pervasive component model, extensive prebuilt application services, and a wide choice of programming language and tools support.

Windows DNA applications have proven themselves in a wide range of circumstances, and the value they represent in the modern distributed computing environment has been thoroughly demonstrated. They have, however, also shown themselves to require careful planning and thorough testing throughout the development process. Avoiding the kinds of mistakes noted in this article should reduce the amount of resources required to produce the kind of Windows DNA application you want. Performance and load testing is unavoidable. Do it in a manner that simulates real-world conditions for your particular application, and you'll be rewarded with an *n*-tier application that works and works well.

References

- www.google.com
- www.wikipedia.com
- www.studymafia.org