

ABSTRACT

The concepts of application development have been rapidly changing over the last decade. Careful consideration is required to utilise the right technology and architecture to maximise your return on investment (ROI). The present technology facilitates the flexibility to integrate the power of SAS[®] software with a wide selection of programming languages such as Java and Visual Basic (VB). Differences between Java and Visual Basic will be discussed. Applications should be structured into separate tiers, hence maintaining the division between the interface layer, the business logic layer and the data layer. Organising conceptual n-tier architecture will result in rapid development, minimum disruption, enhanced scalability and most importantly, will form a sound foundation for future enhancements. Overall advantages will be discussed giving guidelines in choosing the right tool to develop a successful application.

INTRODUCTION

Communication methods have now been greatly improved to enable the integration of applications. Software vendors are providing interfaces that will allow different applications to make calls and utilise each other's methods through a variety of different programming languages. This paper will provide information on implementing a good application development design and highlight some of the key differences between the available programming languages. There are currently many options available to developers and therefore it is important to fully understand all the available choices before embarking on building an enterprise solution. Attention is focused on programming languages such as Java and VB while providing reasons behind why a minimum 3-tier application design should be used when creating applications.

DESIGN ARCHITECTURE

The use of computer applications within industries has changed drastically since the 1960s. In the early years, applications were encapsulated on a single mainframe, which used 'dumb' terminals to give users accessibility to various applications. These were known to be single-tiered. These applications were mainly written in languages such as Cobol, PL1 and FORTRAN. During this time, developers worked within a simple development environment and the skill set required by a developer was very well defined and the operation of the application was relatively easy. Due to the high requirements for security and data integrity, the accessibility to the underlying information was limited to the users.

When computers were made more available to users in the 70s, the architecture was restructured to allow the application to be executed on local machines and to provide users with a user-friendly graphical interface to perform various operations. As computers were networked together and the use of databases increased, the usage and complexity of applications continued to grow. As a result, the application development architecture was radically changing. The previous single-tiered applications were separated into two-tiered applications where the databases handled the data layer. As the complexity of applications grew, three tiered and N-tiered applications became increasingly popular. So what are these tiers?

A typical application will have three layers which comprises of

the presentation layer, business logic layer and the data layer. A single-tiered application will combine all these layers into one single component where a single executable program manages the GUI, the business logic and also performs its own data storage and manipulation. A 2-tiered application usually combines the presentation and the business logic layer into a single component which utilises a database that manages the data layer. Separating all three layers will result in a 3-tiered application. Most 3-tiered applications use middleware components such as RMI, CORBA or DCOM to handle communications. N-tiered applications are developed when there are many distributed objects across many computers where each object has its own data layer.

Tiered Application Design

The single tiered architecture tends to have simple business logic and a modest amount of data. As data volume and complexity grow, databases are used to handle and organise the data layer. Therefore, data drives the need for a 2-tiered application from a single-tiered application. Most applications today consist of relatively complex business logic. The programming code becomes increasingly unmanageable as the business logic is embedded within the interface layer. As a result, the presentation and business logic layers are uncoupled and this formed the basis of a 3-tiered design.

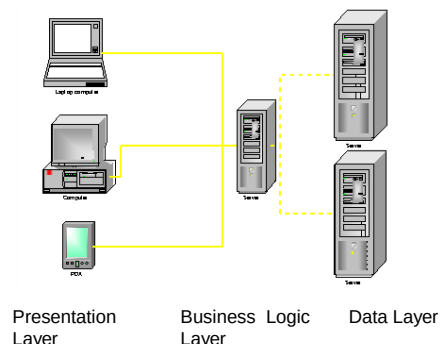


Figure 1: 3-Tiered Application

Figure 1 shows the design of a typical 3-tiered application. Separation of tiers does not necessarily mean separating layers on different machines. It simply means that the layers in the application are "unglued" and dealt with separately even if they exist on one machine.

The separation of layers will allow better organisation and easier manageability as this allows us to focus on each layer individually. The maintenance of this type of application can be performed by larger groups of people who work together to achieve the overall goal. A typical application undergoes several updates over its lifecycle. Therefore modifications have to be performed with the minimum amount of disruption to users.

Adopting a 3-tier design allows developers to rapidly create applications provided that all developers in the different tiers work closely together. Using this design, we can mix and match various different components to gain maximum



efficiency without being tied to one particular software company. Applications can be mainly divided into Rich Client Server applications or Web Browser based applications. The design architecture discussed here can be applied to both scenarios.

SAS can be used in any of the 3 tiers. The current trend places SAS in the business logic and data layers. SAS is well known for its pre-written procedures that perform complex and useful routines allowing the manipulation and transformation of data into useful information. SAS is also efficient in the data storage area and therefore could also be potentially used in the data layers as there are many native data drivers that enable efficient reading and writing of data. For example, connections to SAS data sets can be accessed through any of the following OLE DB data providers.

- SAS LOCAL DATA PROVIDER
- SAS/SHARE® DATA PROVIDER
- SAS IOM DATA PROVIDER

In order to make the right choice between providers, you have to consider the following requirements;

- Locations of the SAS data sets
- Single/Multi user access to SAS data sets
- The requirements for additional complex calculations

The local provider can access data in any directory that is available to a desktop file system without the overhead of Base SAS. It enables data consumers to read and modify data sets directly although the accessibility is only available to a single user at any one time. SAS/SHARE® can access data sets on any SAS/SHARE® Server through TCP/IP connections while the SAS IOM Provider can access data sets on any object server that is available from a desktop. Both SAS/SHARE® and SAS IOM Providers enable multi-user support (record-level lock). For multi-dimensional data, SAS/MDDB® Server OLE DB for OLAP Provider can be utilised¹.

In the presentation layer, there has been a growing trend within the application development community to use Java or Visual Basic. The next section gives a brief introduction of the two programming languages and describes some of the key differences between them.

PROGRAMMING LANGUAGES

Visual Basic

Visual Basic has been in existence for over a decade. It was first launched in the early 90s and its popularity has been growing rapidly due to the ability to develop applications very quickly and easily. Visual Basic is widely known as a Rapid Application Development (RAD) tool and currently, there are large amounts of code resources and skills that are widely available.

Unfortunately Visual Basic programs can only be executed on the Windows Operating System. However, due to the popularity of Windows, developers continue to create applications using VB. Visual Basic for Applications (VBA) is made available within the Microsoft Office Suite and other products such as Business Objects and CorelDRAW. Although VBA is a subset of VB, a reasonable amount of the functionality is present and this continues to encourage developers to program within the VB environment. VBA programs cannot be compiled into a standalone executable file, instead they have to be run within a container. The container can be any one of the Office products such as Microsoft Excel, Microsoft Word and Microsoft Access or other products such as Business Objects.

VBScript, a subset of VBA is used in web pages to make them

dynamic. VBScript is also used in Active Server Pages (ASPs) which allow programs to be executed on the server through the hosted web server. This then produces dynamic pages back to the users based on their selections.

.NET

The .NET technology was launched in 2002. The .NET framework consists of hundreds of classes that are all residing at the same level as the Operating System (OS) layer. Above this layer, A Common Language Runtime layer is provided where VB and other supported languages are compiled into byte code at runtime. These programs can utilise all the classes that are available in the .NET framework to perform various tasks. It is widely accepted amongst programmers that .NET will give you many additional advantages over its predecessor. The .NET framework provides many reusable classes that can be accessed by applications.

As with the transition from 16-bit Windows to 32-bit Windows, the move to .NET will be inevitable in the long term for Window based clients². Developers initiating new applications today should be considering the .NET solution rather than using VB6 to prepare for this transition.

Java

The arrival of Java was known to be one of the most exciting events that took place in the IT world since computers were networked. The language was developed in the late eighties where a pool of companies created the language specifications. Java came at a time when OOP was black art and most programmers were more comfortable to take on Java compared to languages like C++. The main selling point of Java happens to also be its weakest point. Java is interpreted into byte code when it is compiled. The Java Virtual Machine (JVM) then does the interpretation of byte code into machine code at runtime. This makes Java completely portable and places emphasis on the 'write once run everywhere!' concept.

Web-based Solutions: JSP Vs Classic ASP vs ASP.net

The paper will first perform comparisons for web-based applications. There are many publications that have compared Classic ASP to JSP and ASP.NET. Figure 2 illustrates the enhancements brought upon by the .NET framework to the classic ASP applications, which certainly bridges the differences between Java and .NET.

JSP	Classic ASP	ASP.NET
Interface and Code can be separated	Combines Interface and Code	Interface and Code can be separated
Comprehensive Error trapping methods to perform debugging	Limited debugging and error handling	Much Improved debugging and error handling
Good Development Tools	Low/Moderate Development Tools	Good Development Tools for RAD
Easy Deployment with XML files and by using war files	Cumbersome Deployment Procedures	Easy Deployment with XML, ASPX files
Access to entire suite of server side Java libraries	Limited classes available	Full accessibility to all the classes within the .NET framework
Platform Independent	Platform Dependent	Currently Platform Dependent but architecture allows possibility of independency
Compiled into	Pages are cached	Compiled into an

JSP	Classic ASP	ASP.NET
Java Servlets when first called. Then executed by JVM.	but code is interpreted and therefore it is slow.	Intermediate Language (IL) code when first called.

Figure 2: Comparison between Web-Solutions

Java Versus .NET

The table in figure 3 highlights the key differences between the two solutions.

Java	.NET
Platform Independent	Mostly Windows based
Open technology as created with input from many organisations	Relies on Microsoft technologies
License - free	License free but Windows required
Mature > 7years	Relatively New <2 years
Future stability and direction sound as public discussions before standards become final	Future stability and directions controlled by Microsoft
Not Locked-in to one company	Vendor Lock-in
Programming in Java	Programming in VB.NET, C#, J#,C++, etc.
Truly Portable	Portability should increase with time

Figure 3: Java Vs. .NET

The table in figure 4 highlights the key similarities between the two solutions.

Java	.NET
Java Virtual Machine (JVM)	Common Language Runtime (CLR)
Extensible reusable Application Programming Interfaces (APIs) in applications	Extensible reusable Dynamic Link Libraries (dlls) in applications

Figure 4: Java and .NET Similarities

The current trend shows that most Windows based organisations take the .NET route as the preferred solution for easy, rapid client development. On the other hand, Java seems to be the preferred solution for server side development due to better management of security and memory leaks. Java is known to have much richer server-side capabilities such as session bean, message driven beans and clustering while .NET offers a much simpler way of tackling problems all within a fully integrated environment. VB has always been known for the RAD environment. Comparison of code writing between Java and .NET states that the same functionality in an application can be built by writing far less code in VB.NET³. However this can be debated. Java developers including Sun, IBM® and Oracle have indicated that this comparison is invalid as the Java based application was not optimised⁴. Current industry benchmarks show that both solutions can claim leadership in some configuration or scenario. The decision here should be driven by IT infrastructure and available developers skills.

FURTHER CONSIDERATIONS

The following considerations should be taken prior to developing any application.

- Development Criteria
- Deployment Criteria
- Compatibility
- Cost Criteria

- Post Implementation Updates

Development Criteria

The main two areas of the development area are the

- Software Management Features
- Development Tools and Interface

The maintenance of code and structure can be a formidable task especially when there are a group of developers working together on building an application. Developers tend to possess their own signatures when it comes to structuring and writing the code and therefore it is useful to have standard templates that provide a good starting point for all developers. It is good to have a consistent look and feel to each page of code with the appropriate comments, headers and footers to enable the ease of code maintenance. There are many development tools that will allow developers to create and use templates within the build environment. As new items are added or require to be made available to all or a group of program files, templates can be applied across an entire or sub section of a project. There are also ways to add and keep track of a to-do-list on every page so that a project manager can easily keep track of the progress in place. Developers maybe modifying programs concurrently, therefore the chosen tool should enable source code control and version change management. It should be able to support a large development team.

RAD tools such as Visual Editors, wizards, code repositories and predefined classes should be supplied to speed up the development process. References such as documentation, sample code, active development community and the right amount of local training skills will certainly assist in the development process. Developers spend a lot of time debugging code and therefore the tools used should certainly provide useful debugging facility to assist in locating errors and fixing them.

Examples of tools that are widely used include VS.NET for .NET solutions or IBM WebSphere Studio is a popular choice for Java developers.

Deployment Criteria

The chosen language must be forward compatible to ensure the reliability of the application. Platform independency is a strong point to consider as computer hardware tends to change from time to time. Security and scalability issues have to be thought of by considering encryption, load balancing and multi-threading. Ensure that the solution is adaptable to various database support such as SQL support and database connectivity. When building web-based solutions, ensure that the application works across all browsers.

Compatibility

It is important to investigate the compatibility of the solution with your current IT infrastructure. The solution should be integrated well without causing any side effects or disruptions to the running of other programs or causing the slowdown of servers. If some tweaking is required to be done to other existing applications then it is important to investigate the amount of work required and to fully test the integration of all the existing applications with the new architecture.

Cost Criteria

Licensing issues and vendor lock in situations should be considered. The design of some solutions could have side effects on other applications. Therefore, a more thorough design and testing phase may be required to ensure complete stability of the application and to adhere to compatibility issues as discussed in the previous section.



Leaffield, Oxon
OX29 9PG

Work Phone: 01993 878287
Fax: 01993 878042
Email: Vijay.Rajandram@amadeus.co.uk
Web: www.amadeus.co.uk

The concept of metadata can be incorporated into the application design. Although this could slow down the initial development time but in the long run, this will be a significant advantage as users often request changes. This would reduce the process of constantly changing code and also reduce the dependency on hard-core programmers to make minor changes to applications. This will help keep the cost and development time to a minimum in the long run.

Training should be planned and included in an application development project. Applications should be built based on business rules and logic rather than being solely dependent on technology. Adhering to this rule will make it easier for users to learn and use the application.

Post Implementation Updates

As the numbers of users increase, the deployment of applications in client server applications can become increasingly difficult. Deployment of applications should be quick, easy and should cause the minimum amount of disruption. Highly complex configurations in the deployment process should be avoided.

Complete solutions should be packaged well to minimise on the installation and configuration tasks. It is often useful to do some research and obtain timings and performance related issues prior to implementation so that developers can fine-tune the application or the hardware to obtain maximum efficiency from the solution. Load testing tools can be utilised to achieve this.

CONCLUSION

Developers can now mix and match various components to obtain all the advantages into an application. It is important to investigate all the aspects mentioned in this paper prior to building an application as this will assist in creating a well-structured application that is robust, scalable and stable. The choice between Java and VB should be determined by the available skill sets and the existing infrastructure that is currently available. Both languages have their share of advantages and disadvantages and therefore it is important to keep your options open and adopt the best solution as and when appropriate. The architecture to aim for is a design that is relatively impervious to change. The central core of an application should be built around business rules and relationships which tend to change at a much slower rate when compared to technological advancements.

By integrating applications throughout an enterprise, we can contain cost, utilise all the available staff skills, achieve interoperability between applications, reuse of software systems and finally maximise the ROI.

REFERENCES

- 1 Cox, Thomas W., (2000), SAS Institute White Paper, What's Up with OLE DB?
- 2http://www.fawcette.com/dotnetmag/2002_02/magazine/columns/strategy/default_pf.asp
- 3<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/psimp.asp>
- 4<http://www.middleware-company.com/j2eedotnetbench/>

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the author at:

Author Name	Vijay Rajandram
Company:	Amadeus Software Limited
Address:	Orchard Farm, Witney Lane,

© Amadeus Software, 2003.

The Amadeus logo is a trademark of Amadeus Software Ltd, Witney, Oxfordshire, England.

Microsoft products are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.