

Execution Platforms

**The war between
.NET vs. J2EE**

Agenda Outline

- **Preparation (Concepts)**
- **Historical Tour of Execution Platforms**
- **Technical Analysis**
- **The future**
- **What it means for us**
 - **Consumers**
 - **Developers**
- **Conclusion**
- **Discussion/Question Time**

Execution Platform – Huh?

- **First – The Real Thing – an operating system.**
- **Then – Competition – another operating system**
- **Again – Hey, slow down – more operating systems**
- **...**
- **Chaos – OS built by Tom, Dick, Harry and Bill**

In this section, I want to examine the concept of an execution platform. Let me illustrate the point by means of a timeline flythrough of the concept.

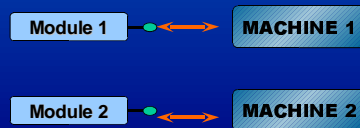
1. The concept of the Operating system was really a turning point for computers. It transformed single-purpose machine into multi-purpose devices. (Think of a pocket calculator compared to your PC).
2. The problem was that, in time, there was a multitude of platforms that provide a world of choice for the users – mainframes, Pcs, Handhelds, Smart Cards, Cell phones, midrange computers
3. Nothing wrong in that – except that the choice is maybe too much. Too many different OS's, on different hardware, with different "standards", for different functions, by different vendor/user communities.
4. Then came the competition – law suits, "mine is bigger than yours" syndrome, "anything but Microsoft..." religion.
5. End result = one holy mess.

Execution Platform

Scenario – An E-commerce product

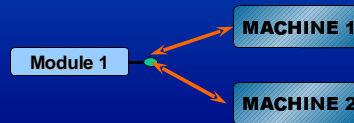
Option 1 Version for each OS

- ✓ cater for all users
- ✗ we miss out a few
- ✗ we don't have the skills/technology
- ✗ it costs too much



Option 2 Single Version – runs everywhere.

- ✓ Cover all uses
- ✓ Single effort and cost



The solution, a single "operating system" that runs on different OS's, on different hardware, with different "standards", for different functions, by different vendor/user communities. With the commercialisation of the Internet in 1994, this was needed like never before.

Let me explain this with an example.

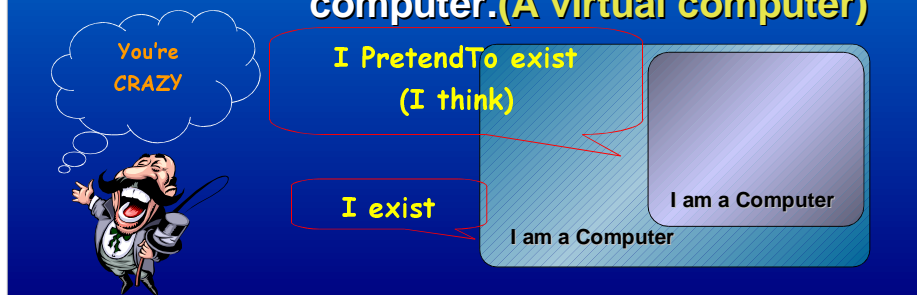
Scenario : You create an E-commerce component that sits on your web site and must be run by a customer before using the ecommerce product. You have no way of knowing if the client is logging in from a Linux, Windows, OS2, Mac., or Cellphone, or any other Internet enabled device.

Option 1 : You can create a separate version for each operating system out there (at great cost and inconvenience to you and the client).

Option 2 : You can use technology that will take a single copy of your code, produce a single executable, and run it on ANY computer, irrespective of OS. Hardware, etc.

The Dream of Portability

- Wanted : A truly portable OS / Platform
- Problem : True portability is not possible
- Solution : A computer that contains a program that pretends its a computer. (A virtual computer)



Whatever the product – a vision was needed - the ability to get programs to run without being concerned about which operating system, which hardware, which drivers, and so on.

Let's look at one way to do it.

Q : If we think carefully, despite all the different type of hardware and operating systems that are available today, what's the ONE thing that's common today.

A : They can run computer software programs.

So, we need a COMPUTER, but we have a PROGRAM.

So why don't we create a software PROGRAM that a computer. In other words, let's create a PROGRAM that pretends it's this wonderful computer.

Since the software can be created to be anyhow we like, why don't this pretend-computer truly portable.

If you understand the problem, then you'll understand the solution.

The solution that we are creating is not a computer. It's not an operating system.

It's an environment that allows you to run programs.

In other words, it's an EXECUTION ENVIRONMENT.

Execution Platforms

- Two significant projects
 - The Java architecture (J2EE)
 - First project to gain widespread commercial support
 - The .Net (read Dot Net) architecture
 - Microsoft product / Framework.
 - Looks important.
- Other significant frameworks
 - Corba

Execution Platforms

Over time, there have been many attempts to solve this problem and create a well designed execution platform. Today, I want to look at two significant projects to address this problem.

- We will look at Java, as it was the first project to gain widespread acceptance as a commercial tool.
- I want to look at the .Net architecture as it is gaining widespread support as a viable alternative to Java.
- The .Net architecture was developed by Microsoft, and judging from the media attention, as well as the things that Microsoft is saying, it looks as if this technology is going to be an integral part of Microsoft's future success.

The reason why I've chosen these two, and indeed why I've labelled my talk platform wars, is because the comparison of the two products is very much in the news these days. The two products seem to be engaged in a low intensity war to see which product is bigger and better.

For us, as people involved in the IT sector, I see this as an important sign of things to come. I feel that these two architectures are going to shape the future of computing, and that the progress made is going to have an important impact on the way we live our lives. It is therefore important to understand the technologies, and analyse their potential impact on the computing environment of the future.

Java – A historical perspective

- ☐ Derived from the "Forest" project.
- ☐ "Compile once – run everywhere" principle
- ☐ Platform independent
- ☐ High level of Portability
- ☐ Java consists of
 - ☐ The Java language
 - ☐ Predefined libraries – very good
 - ☐ The Java Virtual Machine

Java

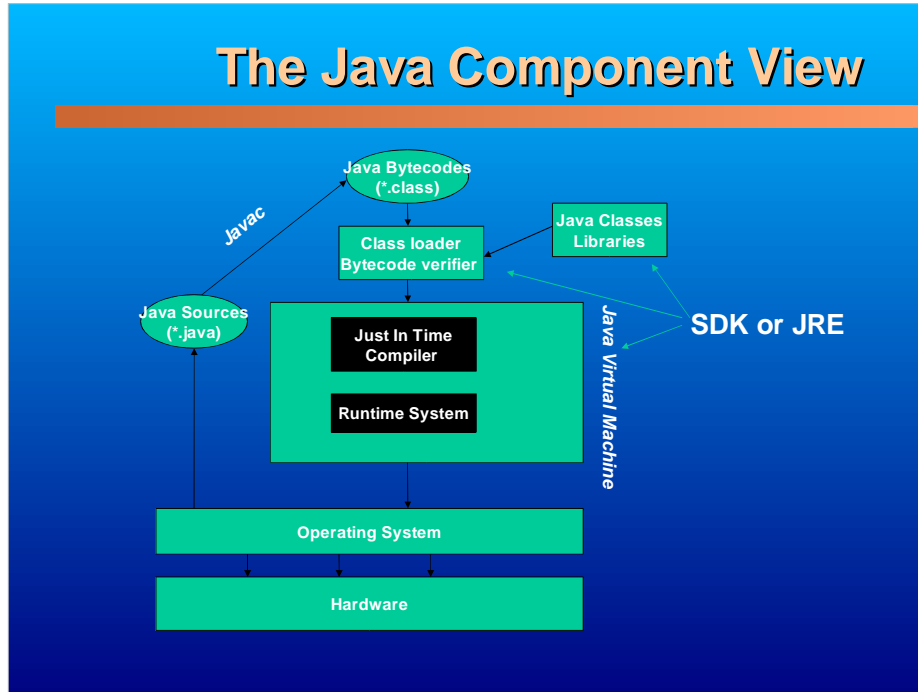
I want to continue my discussion of the concept of Execution Platforms looking at the architecture of Java.

What we know as Java today was actually a rework of the "Forest" project. Java is a vision of a language that you can "compile once, run everywhere". Java was therefore built from the ground up to run on a number of platforms and achieve a high level of portability.

Most people think of Java of a programming language. In fact, Java is a platform consisting of 3 main components:-

8. The Java language itself that programmers use to create programs
9. A HUGE library called the classpath that allows Java programmers to do all kind or weird and wonderful things without much programming.
10. A virtual computer called the Java Virtual machine (or JVM) that executes the JAVA programs.

The Java Component View



The Java Component View

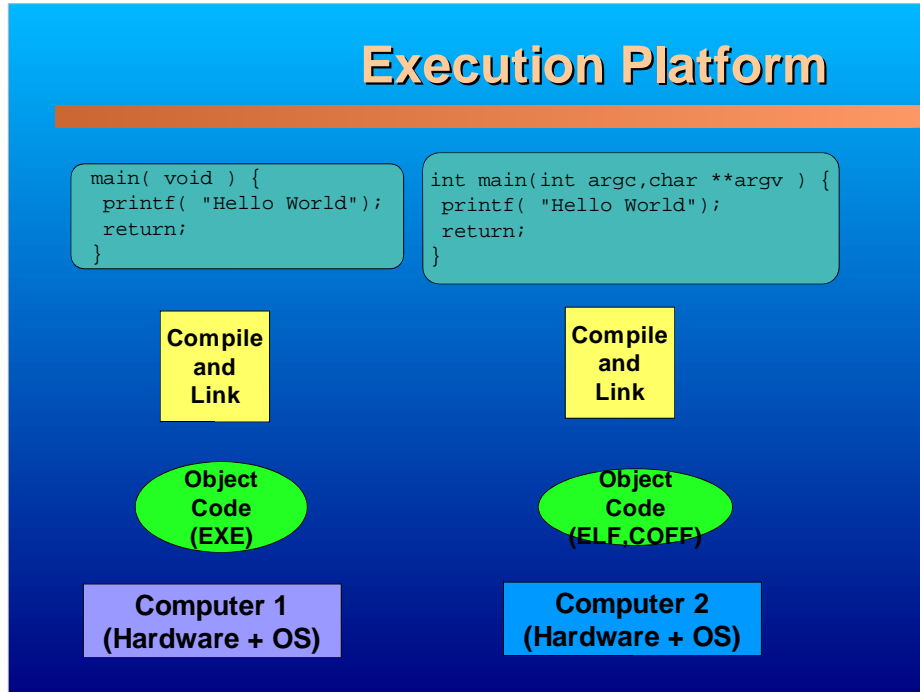
To achieve the portability we are discussing, the designers of Java created a model of a computer that will allow Java to be portable from day 1.

So they designed a "computer" from scratch – complete with CPU / memory and such nice things. This computer, however, was no ordinary computer – it was a virtual computer.

In other words, there is a computer PROGRAM running on your computer, pretending it was a computer. This make believe computer understood special executable code (just like your computer) and ran it. When it needed to do something that it couldn't do by itself, it just asked your computer.

This virtual computer was developed hand-in-hand with the Java programming language, and is therefore called the "Java Virtual Machine", or JVM.

The JVM has built in safety mechanism, for example, it won't just crash, it has security measures to prevent outside programs from accessing your hardware and network.



The Java Virtual Computer

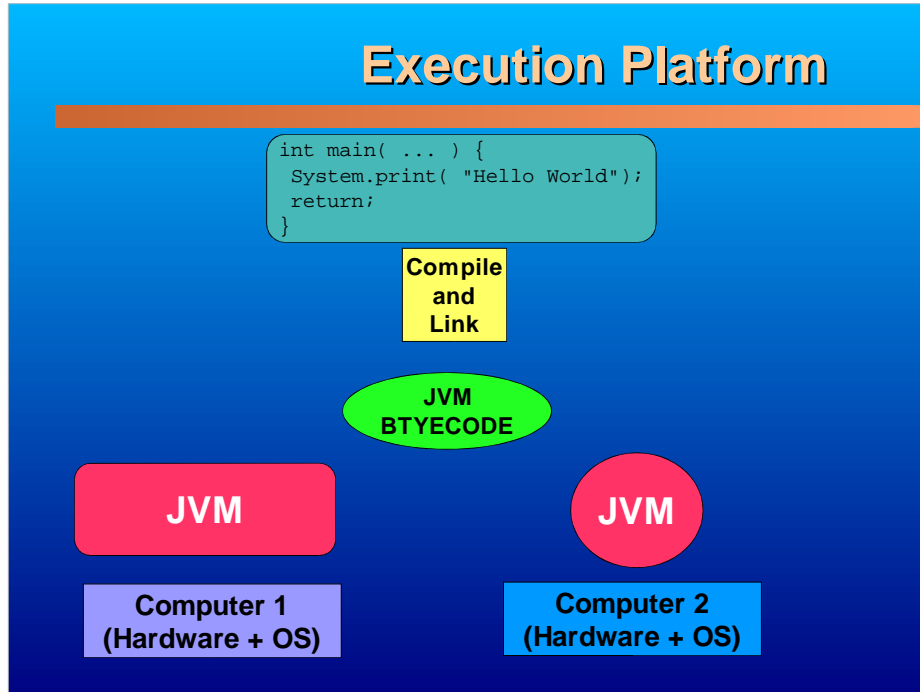
If all this seems too much to handle, let me show this graphically.

In this slide, you can see the normal way of doing things.

Your program "Hello World" is compiled for the target platform, and then run.! An executable program can only run on its target, (say on Windows). To run the program on (say) Linux, you must compile the program on Linux and create an executable in a completely different format, for example ELF, or COFF. This is sometings not as easy to do as it is to say.

As you can see, the "Hello World" program might be different for each platform it must run on. Even low level languages like assembly language are not portable.

It addition, it must be recompiled for each platform it must run on. If the compiler is not available for that platform, you have problems. For example, you probably WON'T get a Delphi compiler to run on a cellphone.



The Java Virtual Computer

In the JVM scenario. The target platform is the JVM. Since the JVM operates the same way on all platforms. Java can have really portable features.

It works like this.

- A programmer writes a Java program.
- The programmer "compiles" the Java program. The compiled program is like an EXE file, except for the following VERY important difference:- the executable program is run on ANY platform. How? Simple – it is not run on just ANY platform – it runs on a platform that can run JAVA executables.

Which platform? The Java Virtual Machine. The Virtual Computer.

Remember that the JVM is just another piece of software running on your computer, just like any other program, for example a spreadsheet program.

If the JVM wants something that a virtual computer cannot do (for example, it needs to access the network), any guesses how it does this? It asks that actual computer.

If you are following me right now. You would have realised that a different JVM is written for EVERY platform it must run for. For example, the JVM that runs on your PC cannot run on your cell phone. So a special JVM must be written that runs on your cellphone.

The advantage of this is that once this is done. The Java program written will run on both your PC and the cellphone!

The J2EE Framework

- Java based Technology Stack.
- JavaPlatform + XML + Web Services
- Web Services
 - Allow applications to share data.
 - Discrete units of code.
 - Each handles limited set of tasks
- **J2EE is a STANDARD**

The J2EE Framework

So far, we have dealt with the historical detail of JAVA. Used by itself, JAVA have had fair success in achieving its goals.

With its cross platform capabilities, Java was ideally suited to manage the aspect of web services.

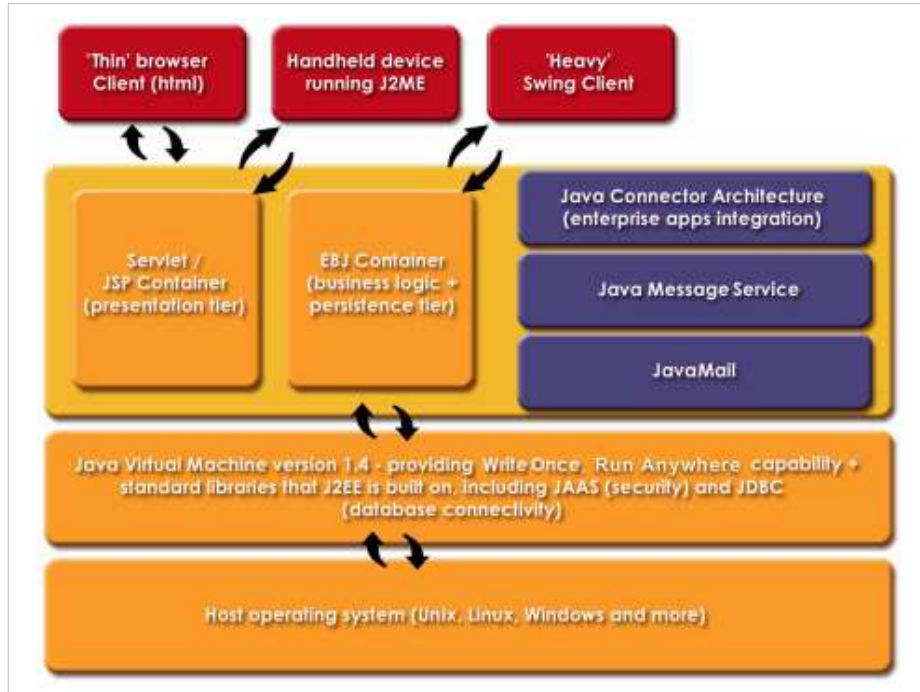
Form this historical perspective, the J2EE standard was formed, AND IT IS IMPORTANT TO NOTE THAT J2EE IS A STANDARD

J2EE (which means the Java 2, Enterprise Edition) but sometimes also called the JAVA 2 Execution Environment) was formed out of the combination of

- the JAVA platform, which is a product
- The XML (extended markup language), which is a protocol
- Web services, which in reality is an extension of web services,

J2EE is build up of layers of technology that interoperate with each other. In the process of allowing the interoperability between all the layers, it achieves a multiplatform standard that can VERY effectively be deployed for the deployment of web services.

Now, I'm not going into the details of the J2EE framework, as it is very complex and will not be covered adequately in the time that we have. However, if you look at the next slide, you will see the tiered approach to the J2EE framework.



The .Net Vision

“ As a result of the changes in how businesses and consumers use the Web, the industry is converging on a new computing model that enables a standard way of building applications and processes to connect and exchange information over the Web. “

Bill Gates - 1999

The .Net Architecture

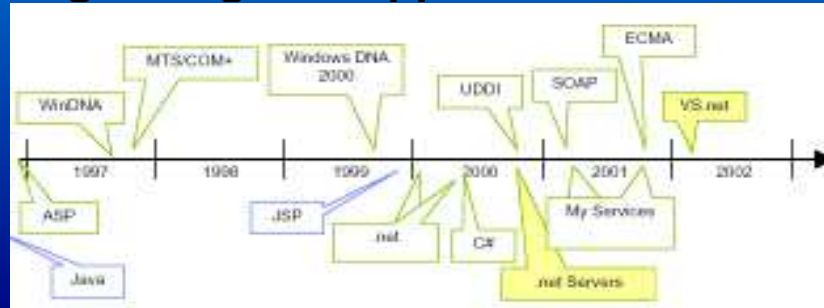
I want to now introduce you to the .Net Framework.

Before I do that, take a look at this quotation from Bill Gates, at the point where the .Net Product was launched. This extract gives a fairly good idea of the Strategy that Microsoft Had in mind four years ago.

If you pay close attention to the parts that I've highlighted, you'll see particular points Of that vision coming through – these keywords provide a summary of what the .Net framework Is all about.

The .Net Timeline

- Initially a new name for Visual Studio Product.
- Change in strategy in response to growing web application use.



The .Net Architecture

The Microsoft vision of .Net is not the same as in 1999/2000, when it was ready to release it for public evaluation.

Initially, .Net was a PRODUCT. It was another name for the Windows DNA product, which was a tool for developing programs and applications for the Internet. It combined new versions of Visual Basic, Visual C, and Visual J into product that could be intertwined into web-enabled and web-centric applications.

A lot of critics say that Microsoft was played catch up at this time. A few years ago, Bill Gates was quoted as saying that the Internet was not important enough for Microsoft to pay sufficient attention to.

Being a late starter, together with the inroads that free operating systems like Linux have made, especially in the area of web usage, Microsoft had to ensure their products were strong enough to be compatible in a fairly lucrative market.

Late in 2000, Microsoft unveiled C# (C-sharp). C# looked like C++, but acted like Java. In Fact, Microsoft came under a lot of fire by critics, saying that Microsoft had stolen the Idea of Java, and is now using it to provide a rival product to Java.

Microsoft extended the gains they made when they introduced .Net Servers, where they basically Packaged a lot of servers under a single marketing product. It became obvious that the .Net concept can be extended to all levels of the web applications framework.

Overview of .Net

- **Multi-language**
 - VB, C++, C#, Perl, Cobol, Fortran ...
 - More to be introduced
- **Uses CLR**
 - Allows many programming levels to be mixed together in one system.
- **Contains well defined library (competes well with the Java library)**

The .Net Architecture

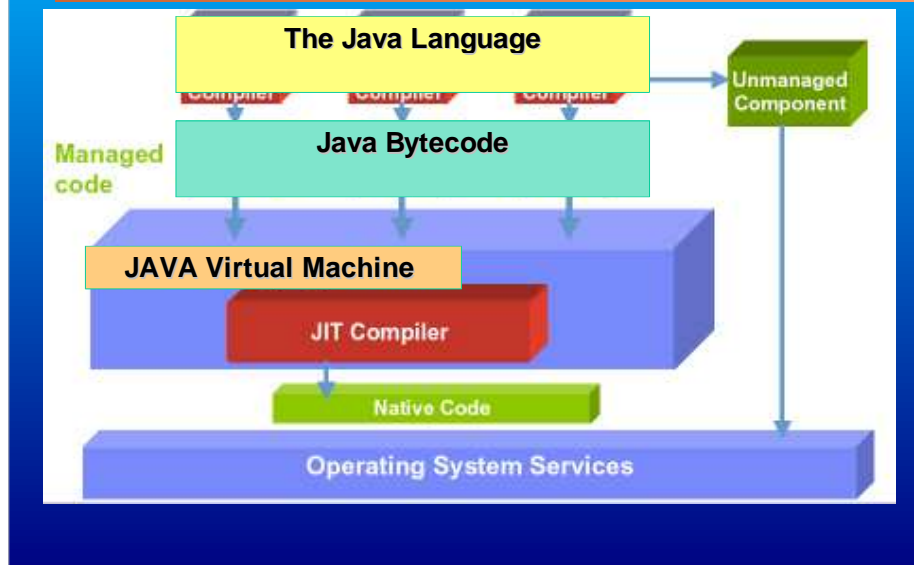
Let's take a high flying view of some the features of .Net.

The framework contains a well defined library of functions and subroutines that encapsulate common and complex programming tasks and provide them in easy to use object oriented libraries.

Certainly, one of the most attractive feature of the .Net Framework is that it accommodates many programming languages. Microsoft Has already hinted that in the forthcoming months, even more languages will be added.

The attraction in the .Net framework design is that the different programming languages can be combined into one heterogeneous application. It does this by using the CLR – The COMMON LANGUAGE RUNTIME. We'll look at this in a bit more detail in the next slide.

Overview of the .Net Framework



The .Net Architecture

Let's look quickly at how Microsoft has allowed many languages to be used in a single application.

What Microsoft has done was invented a low level language that you and I probably won't use. They call this an **Intermediate Language**. In fact, the proper name for the language is **MSIL (Microsoft Intermediate Language)**.

Assume a function is written in VB, and is called from another function written in C++. When you compile the code, it gets converted to MSIL. So you don't really compile the code. You merely convert it from one form to another.

It is not your C++, or VB code that get compiled. The MSIL is passed through to the CLR, The Common Language Runtime, where the code get compiled and run.

In other words, the code is compiled each time it is run, hence the term JIT Compiler.

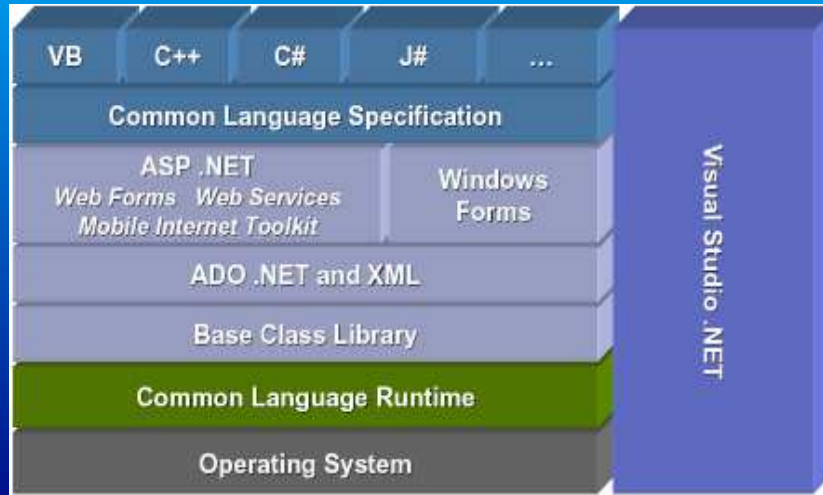
At this stage, I want to draw the comparison between the .Net Platform and that of JAVA.

At the first level, there is a language. In the case of Java there is just a single language.

At the next level, however, the source code is converted to the MSIL. That's no different to the Java source code being compiled into Java Bytecode.

It is the IL, or the Bytecode that is passed to the runtime. With JAVA, that's the Function of JAVA VIRTUAL MACHINE, Or JVM. That's the Virtual computer we spoke At the start.

Overview of the .Net Framework



The .Net Architecture

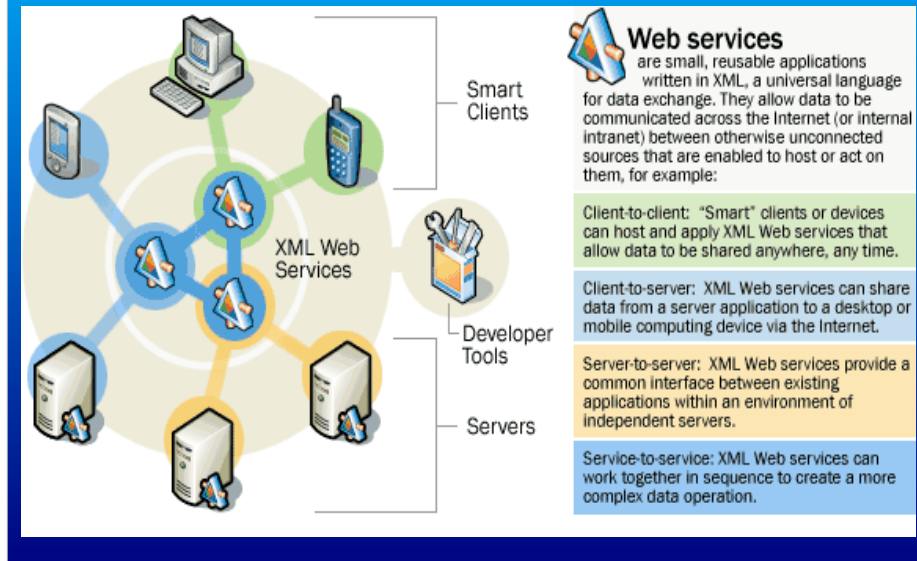
This slide shows the overview of the .Net Framework. What I want you to do here is look for the components that we have already covered, then look at the components that have been filled in to cover the gaps. That's what make the difference between the .Net PRODUCT and the .NET platform.



The .Net Architecture

This is another look at the .Net Framework, showing the tiered approach to Web services development.

XML and Web Service



The .Net Architecture

I've taken this slide directly from the Microsoft web site, because I think it illustrates the basic elements of .Net

You can see from the slide that XML web services play an important part to the .Net Strategy.

Microsoft is targeting the following categories of products:-

h) **Smart Clients**

By enabling .Net technologies into Client computers, for example PC and hand-held computers, those components will have access to web services.

b) **XML Web Services**

By providing the a core set of predefined web services, the technological constraints of creating them from scratch will be eliminated. For example, if Microsoft provides an XML calendar service, cell phone manufacturers might build application to use calendar applications across the web.

c) **Servers**

By embedding .Net into servers like SQL servers and the Exchange Email servers, it will be possible to merge and access those services.

d) **Developer Tools**

Microsoft's development tools , like Visual Studio will be designed for .Net.



Decision Time

The .Net Architecture

The one thing that you should have realised now is that the operating system as we know it is no longer the platform for running applications. Cross platform operation is important. Mobile computing is on the increase.

What choices do we have?

There are a few, but the two dominant alternatives is what we are looking at today.

Businesses will have to start making a few choices. But which one.

Which one ???

- Larger companies will probably support both.
- The challenge is get them to work together.
- Smaller companies will be driven by solution providers.
- The challenge is which crowd to leave out.
- Existing skill / infrastructure is key

The .Net Architecture

For large companies, there will probably be no choice. Because it is fairly common to find a heterogeneous computing environment in large companies, they will probably have to support both technologies. How the two solutions interact with each other will be the proof in the pudding.

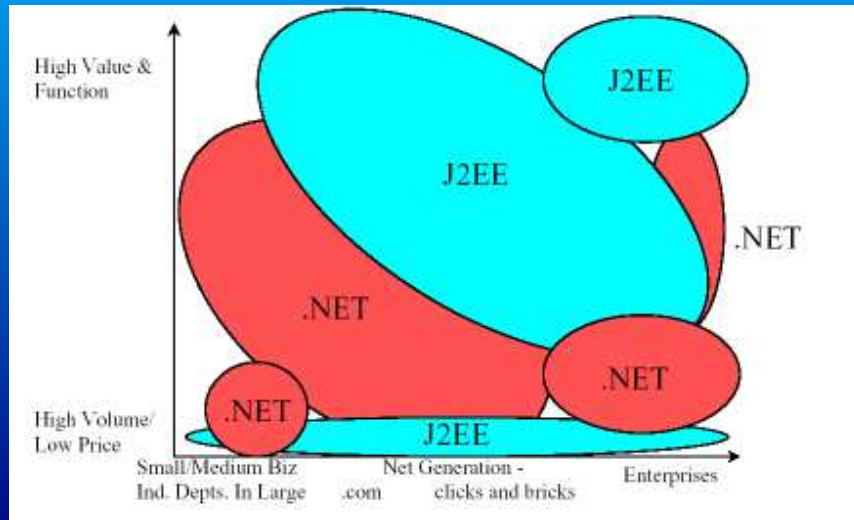
Smaller companies generally rely on solution providers to dictate technology. The choice is then passed onto the provider of solutions. The dilemma these organs face is whether to choose one, or both.

Take for example a company that produces a solution that excludes the Linux platform. There's no telling where Linux will be going in the next five years. Some analysts predict that Linux installations will exceed Windows.

You don't want to miss the boat, so what do you do?

The organisation must be guided by its current level of skill and infrastructure.

Which one ???



The .Net Architecture

I found this interesting slide by American company DH Brown Associates that provide an analysis of the positioning of the two Frameworks in industry.

As you can see, .Net seems to be geared more towards SMMEs and Low prices solutions.

The interesting part is the positioning of J2EE at the lower price end and then again at the higher Price end.

Recall that I told you that J2EE is a standard, not a product. This means that ANY vendor can implement the J2EE framework. There are a number of free or low cost products (for example open source projects) that could account for the lower end. The higher end might give some indication of how scalable J2EE is.

=====

IN THE NEXT FEW SLIDES. I WANT TO EXAMINE THE VARIOUS LEVELS THE PLATFORM WAR IS BEING FOUGHT ON.

Factors of War

Product loyalists

.Net

- architecture is a breakthrough
- new features for old products
- Tiered level

J2EE

- Feel threatened.
- Suspicious of Microsoft
- Supported by anti-Microsoft coalition

Neither

"Let's meet half-way"

Product Loyalists

The first level I want to examine is the product loyalists. We can't discount this group, since they are the most influential in deciding which path to take.

More example, just because I might have studied Java on campus instead of the Microsoft product range, I will assume that Java is the end-all-and-be-all of IS systems development. I will therefore be unwilling to recommend the .Net product, only because I am not familiar with it.

For Microsoft loyalists, the .Net architecture is generally seen as a breakthrough in technology. It has replaced many seeming disparate technologies with a tiered approach to developing. In addition, it has brought many changes to existing products like Visual Basic and ADO that make it more suitable to develop software for today's computing environment, which is largely network-centric.

Java loyalists feel threatened that the software giant will swallow their hard earned skills. So at this stage, the typical cry from Java loyalists is "Microsoft stole that concept from J2EE, and they are marketing it as is they invented it." They see C# as Microsoft's answer to Java.

Java loyalists are assisted by the anti-Microsoft lobby, which we all know is very vocal and strong at this point. Based largely in the open source movement, they are busy planning to counteract the .Net offering by a bigger, better Java platform.

Factors of War

Product age

.Net

- Brand new
- Sufficient time to address inadequacies of competitors.
- Adopt modern Web technology

J2EE

- Mature product
- Certain Web components not provided
- Established

Product Age

If we consider the age of the two Technologies, Java really comes out looking bad.

.Net has a few years advantage on J2EE. It is newer, so it encompasses all the latest techno-bells, whistles and wizardry built into the product. For example, XML and SOAP are inherently built into .Net, whereas the addition of these technologies to J2EE is cumbersome and inconvenient at times.

.Net is newer, and everyone wants to try it to see what it can do. It's the new kid on the block, and it's getting all the attention recently.

.Net has had plenty of time to analyse the shortcoming of competitor products, like Java, and build those into its new product. In addition, it take existing features from other products and do it better – it's easy, because of of these features were developed over a decade ago.

On the positive side, Java has not had much change the way Microsoft has. Some argue it's because the product is a fairly robust and mature product. True, Java has been around for a while now, and has had opportunity to establish and mature its technology.

Factors of War

Platform

.Net

- ▣ Windows and .NET Server
- ▣ Intel architecture
- ▣ Focused skills

J2EE

- ▣ Window, Linux, UN*X, CellPhones,...
- ▣ Use existing infrastructure
- ▣ Use existing skills

Factors of War

Language

.Net

- Many - Use existing skill.
- Non-OO skills to OO (e.g.. Cobol to OO)
- Older code portable

J2EE

- Java Skills required.
- OO

Programming Language

Although the cherry on the top for .Net is the ability to use multiple languages, its must still be done in an object oriented fashion.

There might be a steep learning curve for this.

Factors of War

Price

.Net

- Low entry cost
- High Upgrade Costs
- Bound to certain technology products
- Single Vendor

J2EE

- Low/Free Entry
- Good interoperability provides choices
- Multiple Vendors

**Be guided by the TCO
(Total Cost of Ownership)**

Price

For many organisations, price will be a dominant factor. This is unfortunately a two edged sword.

ENTRY COSTS

=====

The .Net Framework has a low cost. Microsoft has indicated that in time, .Net will be embedded into all their product. While this is an important point that I want to bring up again, what this means is that all the technology will build into your operating system. The problem with the Microsoft strategy is the potential costs of continuous upgrades that is typical of their other products.

The J2EE framework also has a low, or even “no cost” of entry. This make’s it attractive to organisations that are investigating options or “trying out” solutions.

INTEROPERABILITY COSTS

=====

Because of the emphasis on portability of the J2EE Framework, you are not bound to any specific products to make it work. J2EE displays good interoperability with other frameworks and technologies.

Microsoft, on the other hand has designed a suite of products on all tiers of operation. It is obvious that the best option if adopting the .Net approach is to become a “Microsoft Shop”. This could be a costly exercise.

Factors of War

- Performance

.Net

- Unproven in Industry

J2EE

- Historical Data

**Be guided by the
Performance/Price Index**

Performance

Now this is a slide that I didn't really want to discuss. I think the amount of detail says it all. This is a hot point for debates.

The bottom line is that .Net has not matured to a point where it is used at Enterprise Ends. J2EE has, but its use in such applications are not widespread.

Measuring performance is almost impossible with the two Frameworks.

.Net only runs on Windows platforms. This is unfortunately not the platforms that huge organisation use to run their Enterprise solutions.

On the other hand, while the J2EE lobbyists claim that J2EE has excellent performance, performance on Windows is TERRIBLE. Of course, the JAVA crowd complain that it's because the Operating system is in such a terrible state.

Factors of War

Development Tools

.Net

- VS.Net good
- Integrated languages
- n-Tier ASP Model

J2EE

- Many products good choice.
- Free, good products.

**Be guided by the
Performance/Price Index**

Development Tools

The maturity of the Development Tools play an important part in the decision making process.

Here we need to take our hats off to Microsoft. The Visual Studio product is a good product that integrates many languages and technologies into a final product. That's Microsoft putting their money where their mouth is! There is no significant competing product for the .Net technologies.

The J2EE has a good range of products, all supplied by many different vendors. It provides developers with a good choice of development tools with varying degrees of scalability and cost. There are few good free products as well.

Factors of War

- The Benchmark – The Pet Store
- Direct attack on JAVA / J2EE
- Lines of Code (.Net x7 less)
 .Net – 2013
 J2EE - 14174
- Performance (28 times faster)
- The facts
 - J2EE product is unnamed
 - Funded by Microsoft
 - No benchmark used
 - Comments/HTML code removed

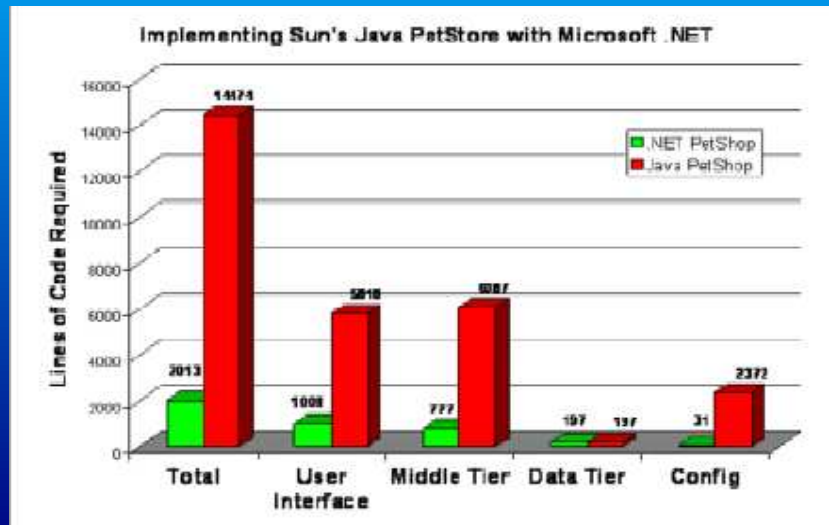
The PetStore Benchmark

I want to break this line of analysis to introduce a controversy that has been attracting significant interest by the Development community recently.

J2EE products is normally provided with a complete E-commerce product called the Pet Store that is used as a teaching tool.

In what is seen as a direct attack on J2EE and in response to performance test done by Oracle, Microsoft performed a rewrite of the Pet Store applications and published the results on its web as a benchmark.

Factors of War



The PetStore Benchmark

Factors of War

- Other comparisons
- Scalability
- Vendor(s) Support
- Transactional Cost
- Learning Curve
- Legacy application Migration

Other comparisons

Factors of War

- **Non comparative factors**
- **Microsoft Vision of .Net**
- **Open Source Movement**
- **Technology trends**
- **J2EE wake-up call**
- **Room for improvement**
- **Alternatives**

The Microsoft Vision

- XML is embedded into all things Microsoft
- Future products all with .Net baked in
- Good marketing strength
- Low entry costs for SMMEs
- Must counter threat from Linux
- Challenge to create robust, high quality products

The Open Source Movement

- Driven by Linux
 - Vision of replacing Windows
- Used more for server type applications
- Low cost for risky projects
- Portable
- Good support infrastructure
- Network/ Internet based from Day One

The Influence of Mobility

- Cellphones / PDA technology merging
- Intel – Centrino Chip
 - Intel is a networking company
 - Centrino chip biggest announcement since Pentium chip 10 years ago
 - Centrino built for mobile computers
 - Network embedded in CPU
- Changes in Wireless Ethernet / Wi-Fi

Other factors

- J2EE wake-up call
 - Slow progress – good market share
- Room for improvement
 - Competition is good
 - Frameworks may redefine themselves in response to developments.
- Alternatives Frameworks
 - CORBA – Notable alternative (Middleware)
 - JBOSS – Java Based. Open source
 - BEA / WebSphere (IBM) – J2EE based

My View of the Future

- **Mobility**
 - Hand held computers will prevail
- **Communication technologies (like phones) will be embedded**
 - Days of cellphone is numbered
 - Computers will have expectation of mobile communication
- **Operating Systems will be irrelevant**
 - All you need is a browser
 - Thin Clients

Challenges to Overcome

- **Improve networking capabilities**
 - Redesign of Internet?
 - Portable IP. In use, but experimental!
- **Virtual Storage**
 - Storage devices are slow
compare 2.8 MHz CPU against 9600 PRM disk
 - Selective storage/retrieval
- **Security Issues**
 - Allows for greater mobility
 - Private networks over Internet (VPN ?)

Challenges to Overcome...(cont)

- World issues (poverty, war...)
 - Technology is for privileged class
- UBUNTU
 - Take a leaf from the Open Source book.

My challenge to you

Find

A SINGLE PRINCIPLE

that drives the computing
environment today

Product Loyalists

If we consider the age of the two Technologies, Java really comes out looking bad.

.Net has a XXX year lead on J2EE, so it encompasses all the latest technobells, whistles and wizardry built into the product.

Java on the other hand has had these bells and whistles added on, sometimes in a very inconvenient manner.

.Net is newer, and everyone wants to try it to see what it can do. Not so with Java.

.Net has had plenty of time to analyse the shortcomings of competitor products, like Java, and build those into its new product. In addition, it takes existing features from other products and does it better – it's easy, because some of these features were developed over a decade ago.

Java has not had much change the way Microsoft has. Some argue it's because the product is a fairly robust and mature product.

Questions

**Perfection is achieved
only on the point of
collapse.**

C.N. Parkinson

DATACRAFT
SOFTWARE CONSULTING



Product Loyalists

If we consider the age of the two Technologies, Java really comes out looking bad.

.Net has a XXX year lead on J2EE, so it encompasses all the latest technobells, whistles and wizardry built into the product.

Java on the other hand has had these bells and whistles added on, sometimes in a very inconvenient manner.

.Net is newer, and everyone wants to try it to see what it can do. Not so with Java.

.Net has had plenty of time to analyse the shortcomings of competitor products, like Java, and build those into its new product. In addition, it takes existing features from other products and does it better – it's easy, because some of these features were developed over a decade ago.

Java has not had much change the way Microsoft has. Some argue it's because the product is a fairly robust and mature product.