WHITE PAPER

December 1997

Compaq Computer Corporation

Compaq Java Positioning

In this white paper, Compaq identifies the standard and proprietary Java technologies, identifies vendor positions, and describes the applications and impact of Java technologies. The white paper also clarifies Compaq's position and shows how Compaq's approach results in the strongest, industry-standards-based networked computing solutions.

CONTENTS

EXECUTIVE SUMMARY

One of the most significant trends of the past few years has been the accelerated movement towards industry-standard network computing – resulting from the convergence of mainframe computing, open systems computing initiatives, standalone desktop computing, and communication protocols. Along with other significant technologies (Browsers, HTML, TCP/IP, SSL, etc.), Java is yet another technology furthering standards-based network computing.

Java has emerged as a significant set of technologies for networked computing. Java and associated technologies are affecting the dynamics of software and hardware development. However, the marketing hype around Java has blurred the precise significance of the technology and confused IT professionals and customers as to what they can actually expect from Java. Java was developed by a small team at Sun Microsystems headed by James Gosling, originally designed as a programming language for programming consumer electronics. Consumer electronics require a small, reliable, architecture-independent language. Sun realized that an architecture-neutral language like Java would be valuable for developing Internet-based applications because they would execute on all different types of computer systems. The design goals of Java were well suited for Internet programming. Even though Java's early use was creating more lively and interactive web pages, developers have committed themselves to Java as a programming language for developing complete business applications within the Internet environment. The success of Java is based on it being a simple, distributed, architecture-neutral, object-oriented, multithreaded, and dynamic language.

Java represents a set of technologies classified into two categories:

1) The *Java language* is a compact, C++ like, object-oriented language that is compiled into a platform-neutral object code (applications).
2) The *Java runtime environment* contains technologies to load and execute (Java Virtual Machine) Java applications. The runtime environment can be bundled with an OS, a browser, or an application.

Java has also been used to implement technologies that improve software distribution and management for networked clients. Although typically associated with "thin" clients, the technologies used to improve manageability are being incorporated into all clients (PC, workstation, etc.). Different vendors have promoted and hyped elements of Java as it serves their purposes. Sun, Oracle, IBM, and Netscape are a few of the vendors that are positioning Java as an alternate "platform," replacing those platforms that exist today. Many vendors are using the "Java" name in their products mainly to leverage the marketing hype of Java (e.g. HotJava from Sun is just another browser). Many have promoted their proprietary/closed implementations as though they were the standard Java technologies. Sun has proposed the 100% pure Java initiative to promote certification of Java for cross-platform compatibility. In addition, many vendors have over-sold the benefits of Java technologies and contributed to the market's confusion.

COMPAQ

As an open standard and platform-independent programming language, Java will be widely used for developing Internet/intranet, as well as traditional applications. Java programs will enable access to legacy applications from a variety of client platforms. Java will also be used to distribute software and manage client upgrades. For networked applications, Java enables vendors to develop programs which reside on a server and can be downloaded and executed anywhere, on demand. Compact, platform-neutral Java programs facilitate development of new thin-client devices, but Java alone is not a sufficient factor for selecting a computing platform. In fact, Java in conjunction with traditional PCs or workstations is a better alternative for mainstream business computing.

Today, the Java environment is lacking in functionality and maturity when compared to industry-standard operating systems. Java application performance varies widely under different OS and browsers. Also, Java programs encounter many incompatibilities and they do not run identically in different environments and configurations. In fact, Java class libraries provided by different vendors may cause the divergence of Java applications based on the Java VM being used.

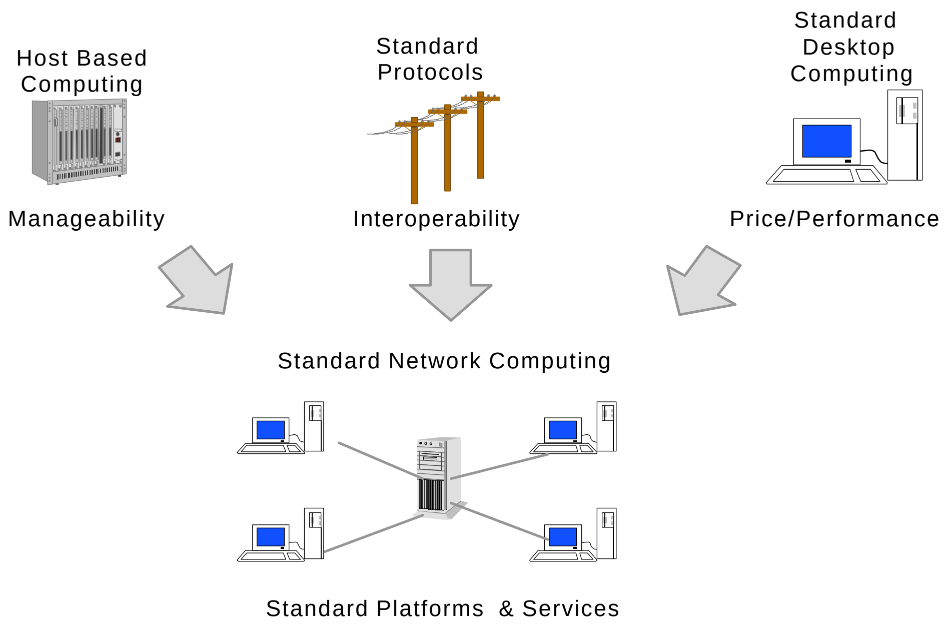
Compaq views Java as an important technology for rapid development, deployment and management of applications. However, Java will not replace the OS, instead, the OS will extend its functionality to incorporate Java. Java does not minimize the importance of the underlying core platform. Performance, application availability, standards, price, installed base, and compatibility are equally important factors driving the choice of platforms.

With incorporation and integration of Java into Windows, the PC (& NetPC) continues to be the best standards-based platform for network computing.

Compaq will continue its approach of providing the best industry standard platforms for the development and hosting of Java as well as traditional applications. Compaq will maintain its commitment to standards-based computing, providing high performance systems with practical, integrated, and compatible solutions. Compaq's approach is one of standard-based network computing, different from other vendors, who may pursue a single vendor/proprietary network computing approach (e.g., requiring servers from the same client vendor).

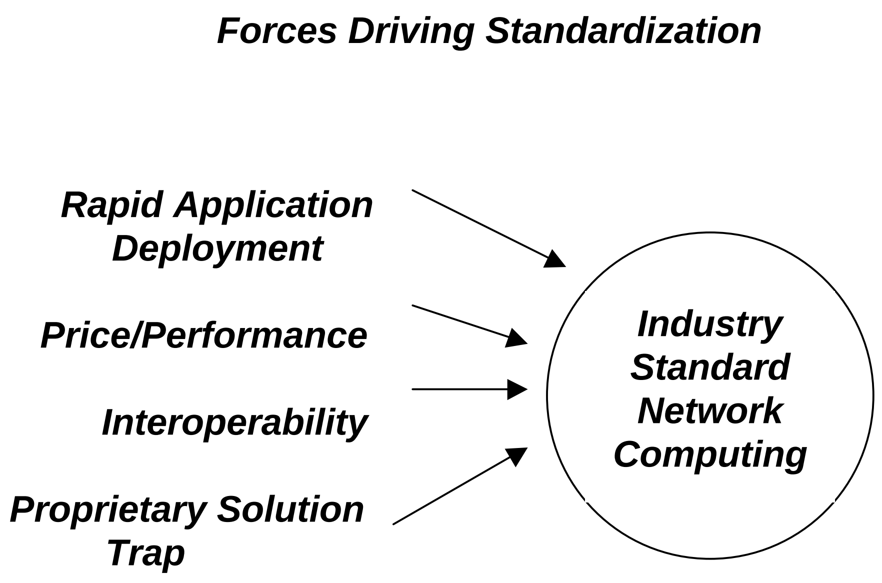
BACKGROUND

The industry, driven by business demand for the ability to manage change in a fiercely competitive environment, has demanded and embraced “open systems” based on standardization. The single vendor, proprietary, total solution has been replaced by solutions based on standards, provided by multiple hardware, software, and service vendors. The industry has witnessed the evolution of a standard computing architecture based on standards that were derived from, and have evolved from the personal computer. The industry is also witnessing the evolution of standards in software, evidenced by the standardization of protocols, GUI, document format, and operating systems. The progression is continuing through its next evolution – the evolution of industry standard network computing.



Several factors are driving this approach to industry standards:

- The ability to rapidly deploy and re-deploy applications based on ever-changing business requirements.
- Capability to deploy applications and services through a standard, network-centric environment where underlying infrastructure can be re-used without overhaul.
- The need to provide access to information anytime anywhere to anyone.
- Interconnectivity and interoperability to integrate multiple, heterogeneous platforms.
- Freedom to choose solutions based on quality and functionality and avoiding dependence on single vendor.

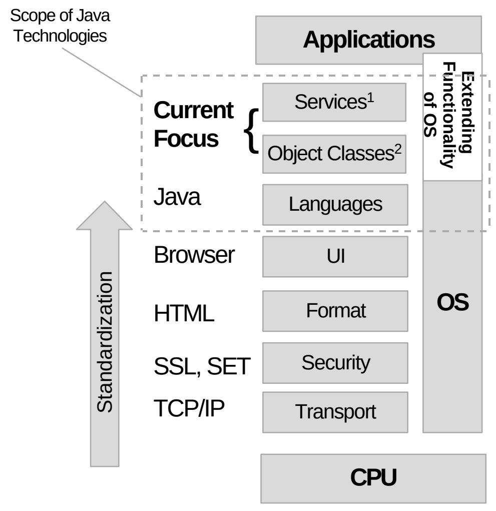

Forces Driving Standardization
Rapid Application Deployment
Price/Performance
Interoperability
Proprietary Solution Trap
Industry Standard Network Computing

The push for industry-standard network computing is occurring through multiple layers of computer architecture. Superior price/performance has made the standards-based x86/Windows PCs the computing platform of choice. Ethernet has become the standard network while the Internet's protocols (TCP/IP and UDP) are the basis for network communications today. Many of the Internet's core services (e.g., FTP, NNTP) form a basis for standard network services. Secure Electronic Transactions (SET) is poised to become the standard for secure E-Commerce transactions. HTML has paved the way for a standard file format for networked documents and the browser-based GUI is evolving to become the standard user interface for information and application access.

Java is one more important component in this move towards industry-standard network computing. As a simple, object-based language, Java will facilitate the development of new network-centric applications. Also, almost all operating systems will integrate Java's Virtual Machine (VM) and provide the capability to run Java applications (residing on a local disk or on a server) on their platforms.

However, a few vendors are linking the "Java" name to proprietary implementations (e.g., the JavaStation and the JavaOS). Given these confusing messages, customers should be careful to separate Java's industry standard implementations and true potential from proprietary implementations and vendor hype.

With the standardization of lower layers, the industry focus has shifted to develop standards for networking and OS services. Higher level protocols and service frameworks are competing to become standards in areas such as messaging, directory, data access, management and software distribution. In addition, competing object models (e.g., CORBA, DCOM) are vying for the dominance for object format, APIs, and inter-object communication standards. Finally, other emerging network services such as real-time multimedia and telephony also lack agreed-upon standards. Establishing open standards in these areas will be crucial to the overall robust development of industry standard network computing.



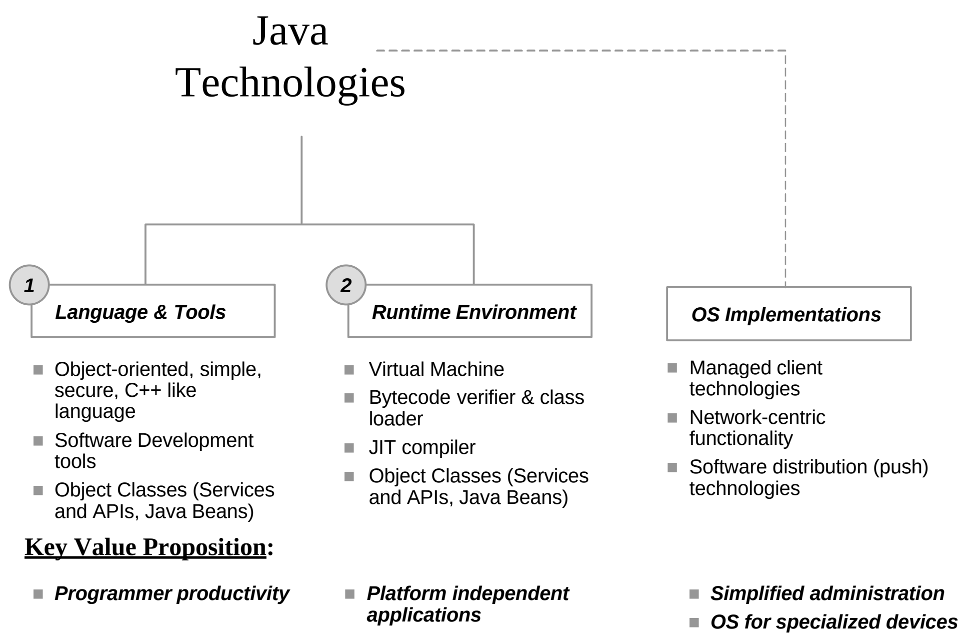
[1] *Includes directory, messaging data access and presentation, management services, push services etc*

[2] *Includes class libraries, APIs and object models used for inter-object communications*

Current operating systems have evolved to encapsulate standard services and offer enhanced functionality. Operating systems such as Windows NT include all the core components (TCP/IP, browser and GUI) needed for industry standard network computing. Microsoft Windows is the de facto standard in this area. Windows' application availability, price/performance, ease-of-use and support for standards are currently unmatched by other computing platforms. Windows is also the leading platform for Java applications because more competition among ISVs has resulted in better (faster, compatible) Java environments.

JAVA TECHNOLOGIES

Java technologies can be grouped into two broad categories – 1) language and object classes, and 2) runtime environment. In addition, Java has been applied in specific operating systems to improve manageability. This third group, OS implementation, is only weakly linked with the core Java technologies.


Java Technologies
1
Language & Tools
2
Runtime Environment
OS Implementations
Object-oriented, simple, secure, C++ like language
Software Development tools
Object Classes (Services and APIs, Java Beans)
Virtual Machine
Bytecode verifier & class loader
JIT compiler
Object Classes (Services and APIs, Java Beans)
Managed client technologies
Network-centric functionality
Software distribution (push) technologies
Key Value Proposition:
Programmer productivity
Platform independent applications
Simplified administration
OS for specialized devices

Java Language and Tools

Java is a simple, C++ like, object-oriented, network-aware, and platform-independent development language. Java is an interpreted language. Java is attractive to programmers because it increases their productivity and if applicable, enables them to write programs only once (rather than modify for each platform).

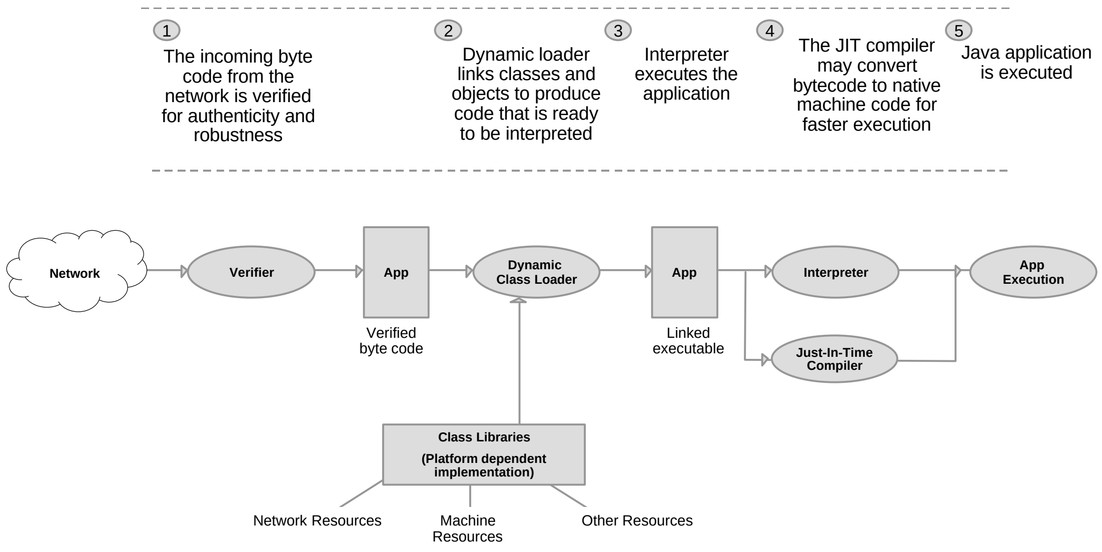
Java's class libraries are critical to Java's development. Class libraries contain objects that implement a platform's core display, network, file and other services so that developers are freed from platform specific details and can focus on core application functions. Like other development languages, there is no one standard class library; rather there are competing implementations of object classes. Currently, Microsoft's Application Foundation Classes (AFC) and JavaSoft's Java Foundation Classes (JFC) are the leading implementations, however, as of this writing JFC is a work in progress, and is not yet being implemented.

JavaSoft has also proposed a new object framework, JavaBeans, for Java objects. JavaBeans specifies the format, introspection, interface, and communication interface properties for Java objects. JavaSoft has also proposed Remote Method Invocation (RMI) interface for communication between Java objects on different systems. JavaSoft's RMI will *compete* with other schemes proposed for inter-object communications by other existing object-frameworks (CORBA and DCOM).

Today, only the Java language has been widely adopted as an industry standard, and Sun has proposed moving the language specifications to the International Standards Organization (ISO). However, the standardization process has just started and currently, Sun is awaiting the vote of ISO members on its request to become a submitter of publicly available specifications (PAS). JavaBeans and RMI technologies are currently controlled by JavaSoft.

Runtime Environment

— Execution of a Java Application —


1 The incoming byte code from the network is verified for authenticity and robustness
2 Dynamic loader links classes and objects to produce code that is ready to be interpreted
3 Interpreter executes the application
4 The JIT compiler may convert bytecode to native machine code for faster execution
5 Java application is executed
Network
Verifier
App
Verified byte code
Dynamic Class Loader
App
Linked executable
Interpreter
Just-In-Time Compiler
App Execution
Class Libraries (Platform dependent implementation)
Network Resources
Machine Resources
Other Resources

The Java runtime environment consists of a bytecode verifier, a class loader, a virtual machine, and an optional Just in Time (JIT) compiler.

When a Java program is loaded onto a platform, the verifier runs a basic set of virus and authenticity checks to ensure the applet is a safe application segment. Java programs consist of only platform independent binary code. The class loader links this binary code with appropriate class library objects. Then the Java VM (interpreter) executes the program providing necessary interface and translation for the native platform services. Alternatively, the optional JIT compiler may translate the program into code for the native OS and CPU, significantly increasing the speed of execution. The Java VM is implemented in software. Although the Java environment can be bundled with an OS, a browser or a standalone application, the trend is for it to be integrated within the OS. All major OS vendors will integrate the Java VM in their platforms.

The Java runtime environment implements a “sand-box” around downloaded Java programs, preventing it from accessing disk and other hardware. This approach results in improved security, but also limits functionality. JavaSoft is likely to change its approach to one based on user authorization in which an explicit permission of user will be required for the programs to access key platform services. This “user authentication” or “trust” based philosophy is similar to one proposed by Microsoft in the security framework for ActiveX components.

OS Implementations

The third group consists of products that incorporate technologies for managing networked clients. These products are not part of the core Java technologies. One such product is Sun's new operating system, JavaOS, for thin clients. JavaOS's association with standard Java technologies is weak. In fact, Java technologies improving client manageability are being integrated into most conventional operating systems.

The JavaOS has a very small footprint, but integrates the entire Java Runtime Environment (VM, Base and Standard Extension APIs). Like a conventional OS, Java OS offers a layered architecture containing an OS Kernel, networking, graphics, and windowing classes. Its small size and network abilities may make the JavaOS a candidate for a variety of thin, network-dependent clients and embedded devices (e.g. NCs, application-specific clients, cellular phones).

However, JavaOS does have compelling disadvantages versus a conventional operating system. In fact, JavaOS does not support existing applications or programs written in languages other than Java. Services provided by JavaOS are limited and the OS environment is immature. Other major computer vendors (IBM, HP, Microsoft, and Oracle) have not shown significant interest in JavaOS.

Sun has also proposed a hardware specification, picoJava, that will embed the Java VM in hardware and provide performance boost. It is not clear whether a dedicated hardware chip is needed to speed-up Java programs when an industry-standard CPU in combination with a JIT compiler may achieve similar or better results. Unlike Java VM, picoJava is not supported widely by the leading vendors. Intel, Microsoft, IBM, and other vendors are making independent efforts and are taking different approaches to increase Java application performance.

JAVA APPLICATION AREAS

Although Java technologies can potentially be used for a wide range of computing applications, it will have the largest impact on network (Internet/intranet) application development and client applications. Java will also be used for selected server applications. Currently, actual applications using Java are limited and a variety of issues in each area must be resolved before Java applications become widely deployed.

Internet/Intranet Applications

Java's portability is attractive for network applications. Java programs can reside on a server and can be downloaded and executed on a wide variety of platforms attached to the Internet and corporate networks. Java-based interface programs can enable access of legacy applications from any standard client platform. Java programs can also be used to add interactivity to a static web page. Java programs are used for form input, data validation, delivering multi-media, and animation. Java's compactness makes it easier to download these applications over bandwidth-constrained networks. Finally, its potential ability to dynamically load and link objects from anywhere on a network enables development of distributed applications.

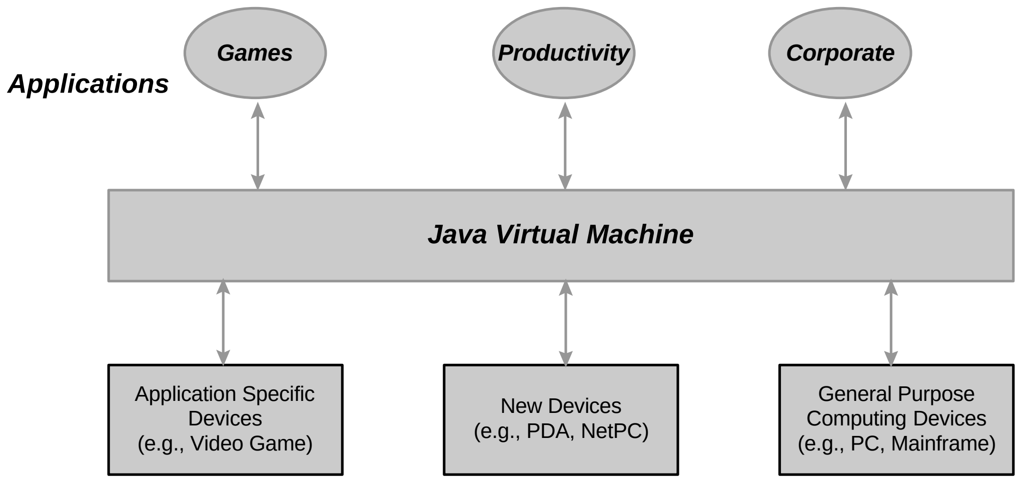
However, there are alternative technologies (HTML, DHTML, ActiveX, CGI , C++, JavaScript, etc.) which may be more appropriate in some circumstances. While Java has been hyped tremendously, there are other factors that a developer must consider. Existing code base, programmer expertise, availability of suitable tools, performance, support, and target platforms are some critical factors that a developer must weigh before making the final decision.

Currently, Java is used relatively infrequently compared to HTML and CGI, which now underlie the vast majority of web sites. HTML provides an especially useful technology alternative for the quick access of information anytime from anywhere. JavaScript (a scripting language, typically embedded into HTML programs, that does not require a developer to understand complexities of object-oriented programming) can be used to link objects and programs. JavaScript is predominantly used by Netscape, IBM and Sun. Other vendors also provided scripting languages. In addition, alternatives such as ActiveX, HTML, and Visual Basic may be more appropriate.

Client Applications

Java technologies have three specific application areas on the client side – 1) development of platform-independent client applications, 2) distributing and maintaining software on client systems, and 3) development of new clients.

Cross-platform applications written in Java are attractive to software developers because they avoid the porting efforts necessary for different platforms. However, testing, verification, and benchmarking on multiple platforms will be required. With its dynamic downloading and linking capabilities, Java is also likely to promote more component-oriented software. Vendors may begin developing Java-based software broken into smaller objects, based on the actual functionality required by specific groups of users. In this scenario, users will be able to download only the part of an application they actually require, rather than having the whole program installed and running on their device.


Applications
Games
Productivity
Corporate
Java Virtual Machine
Application Specific Devices (e.g., Video Game)
New Devices (e.g., PDA, NetPC)
General Purpose Computing Devices (e.g., PC, Mainframe)

Platform independence, network awareness and compact size makes Java a key vehicle for distributing, updating, and maintaining software for a large audience. This feature has applicability in developing more managed clients (thin or traditional). Additionally, push technologies (developed in Java) can be used to configure and update client software from a centralized server. As previously mentioned, the “sand box” implementation of Java prevents Java applications from accessing disks. The push technologies will require JavaSoft to change its approach to one based on user authorization in which explicit user permission will be required for the programs to access key platform resources.

The third application of Java is in the development of new client devices. These devices may emerge on a continuum ranging from a PC to a hand-held video game. Application-specific devices can use the Java VM to broaden their functionality beyond their original specific task. For example, a cellular phone, enabled with the Java VM, can run a Java program that communicates with a PC. To this end, JavaSoft has proposed a number of device class-specific Java APIs: the PersonalJava API (for set top boxes, video games, PDAs, web TVs smart-phones), the EmbeddedJava API (for microprocessor run devices such as faxes, network switches, phones), and the standard Java API (for general purpose computing).

A lot of press has been given to development of “thin” clients (NCs) and its potential to replace traditional PCs. Although Java provides the enabling technology, key business issues still need to be resolved before alternate clients and devices become popular. Critical issues for the new platform include price/performance, flexibility, backward compatibility, vendor support, interoperability, and open standards. Traditional PCs have superiority in all these factors except manageability. With development of Zero Administration Windows (ZAW) in upcoming Windows NT 5.0 and Zero Administration Kit (ZAK) for Windows NT 4.0, the PC platform provides desired manageability.

Today, the PC and NetPC exhibit the highest performance for Java applications and the greatest compatibility with open standards. When compared head-on with the NC, the NetPC platform comes out ahead or equal on all benchmark criteria.

	NetPC & PC	**NC**
Support for Java	Yes	Yes
Centralized manageability	Yes	Yes
Industry standard processors	Yes	Maybe
Industry standard OS	Yes	No
High Performance for Java applications	Yes	Maybe
Support for most business applications	Yes	No
Large vendor support (peripherals & accessories)	Yes	No
Multi-vendor interoperability	Yes	No

As the direct comparison shows, a traditional PC (and its derivative NetPC) is still the best client for business applications. Compaq has analyzed these critical customer issues and has taken a leadership position in developing the NetPC platform.

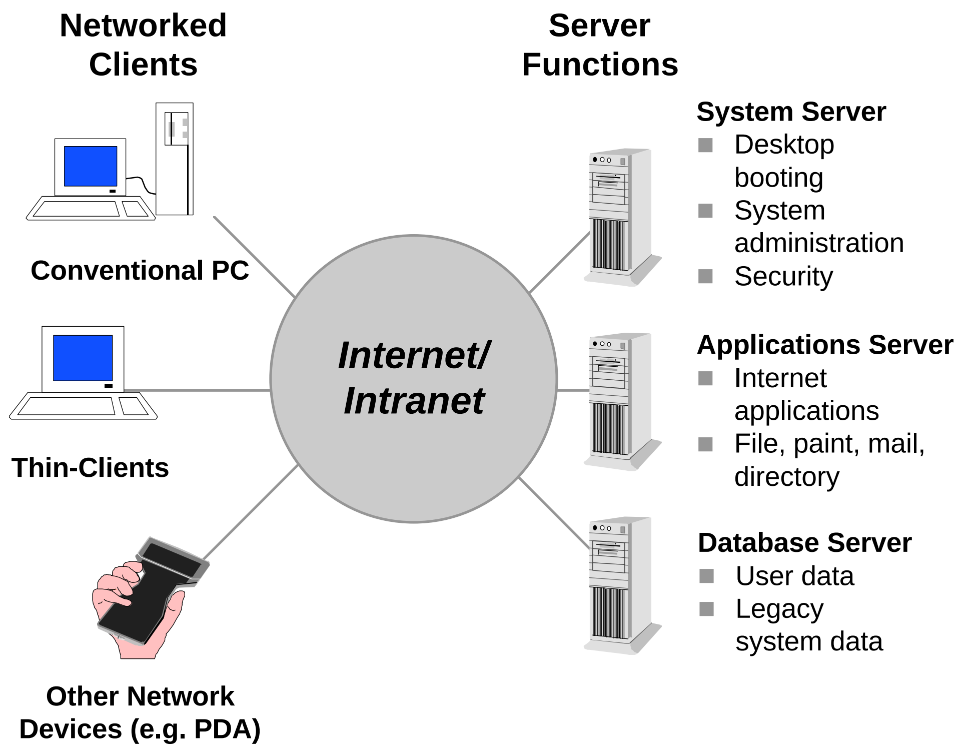
Compaq is focusing on providing the best industry-standard platform for computing by providing compatibility and integration for all traditional and Java based applications. Compaq systems integrate the Java runtime environment and provide a high-performance platform for all Java applications. Compaq systems also incorporate new technologies (Java & others) to reduce Total Cost of Ownership (TCO) and improve enterprise network manageability. Compaq is engaged actively in these efforts through its NetPC and system management software initiatives, such as WBEM (Web Based Enterprise Management).

Server Applications

Java also has potential applicability and implications for server applications. Java applications on the server side include:

- Enabling a single server to support a broader variety of clients – terminals, NCs, NetPCs, and PCs;
- Developing Java and browser-based server management utilities;
- Leveraging Java APIs for commerce, databases, security, and middleware to interact with other server applications;
- Writing non-performance critical application modules in Java to enhance portability;
- Improving performance, functionality, integration, and security of applications by replacing CGI scripts with Java code.

— Increasing Role of Servers —



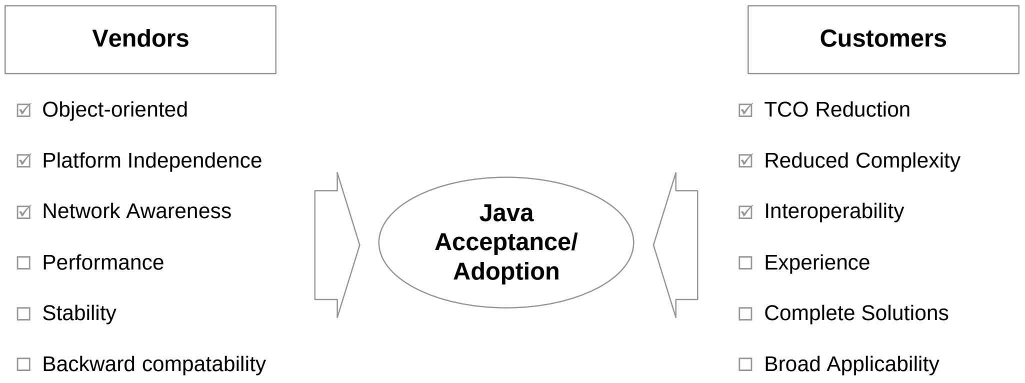
Overall, Java will increase the need for performance on the server side. With an increasing number and variety of network-dependent ("thin") clients, the functionality needed on the servers will increase. Also, the interpreted nature of Java, object orientation, and distributed media-rich applications will fuel demand for greater total computing (client + server) and communications (bandwidth) power for servers.

Significant hurdles prevent Java from being used on server applications today. Performance is a critical factor for servers. Current Java applications are often three to four times slower than applications written in C++. On the other hand, Java applications are typically faster and more functional when compared to those written using CGI scripts. Also, Java APIs are still evolving and many have yet to be commercially implemented. Java class libraries are in their early versions and not mature. Finally, major issues remain around interoperability and the installed base.

Given all of these issues, Compaq, with our partners, will focus on delivering key Java applications deployed on our open platforms. Compaq servers will provide the best combination of price/performance, compatibility, and openness to run Java applications along with traditional applications.

Market Acceptance of Java

While vendors have enthusiastically and universally embraced the Java language and its runtime environment, the Java class libraries (platform services) and object technologies are major areas for competing approaches.



Java has earned solid ISV support as a language. ISVs using Java have noticed marked improvement in programmer productivity. Major independent software vendors are committing significant resources to writing networked applications in Java. In addition, most OS vendors are integrating the Java runtime environment into their platforms this year. However, their enthusiasm for Java is tempered on several fronts. For the most part, ISVs are limiting Java's use to new applications or selected modular functional blocks of existing applications. There are no compelling reasons for ISVs to completely re-write their current applications in Java. Also, incremental improvements are best written in the original language of the program. ISVs are also concerned with issues around performance, stability and interoperability of the Java approach.

On the customer side, enterprise IT managers share vendors' interest in Java but have limited experience and understanding of Java technologies. Customers are intrigued by Java's potential to reduce their overall TCO (partially by using well-managed clients) and improve manageability of their network. However, only a few enterprises have begun using Java in earnest. Most customers have only focused on using Java to improve access to their Intranet/Internet content. Most enterprises are skeptical and are taking a "wait and see" approach until vendors produce complete turnkey solutions using Java. Their strongest desire for Java is for Java products to deliver superior performance, run reliably at a lower cost, provide universal accessibility, integrate seamlessly with their existing systems, and improve their ability to manage the network.

In this market environment, Compaq will continue to focus on making Java technologies accessible and easy to use for its customers. Compaq has been integrating management solutions into its systems for many years. Compaq will continue its leadership by selectively incorporating Java in its systems to lower TCO, improve manageability, and enhance compatibility.

EVOLUTION OF JAVA

Java's Future Role: Application Platform vs. Development Language

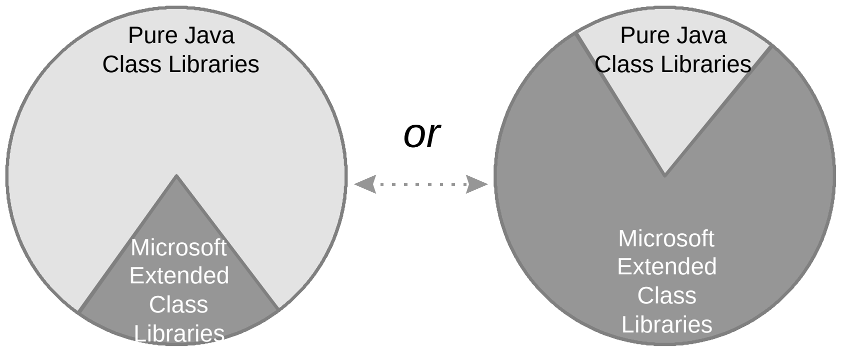
One of the key issues being debated in the industry today is what role will Java play in the future: an application platform or a development language. Sun, IBM and other vendors are trying to position Java as a platform that replaces all existing platforms. Meanwhile Microsoft and other system vendors view Java as an important development language, but not a complete platform for applications. Based on their respective visions, industry vendors are pursuing different strategies for the development of Java class libraries and associated object models.

Java Class Libraries and Extensions

Development of Java class libraries and APIs is the central focus for Java tool providers today. To fulfill its "write once, run anywhere" promise, Java must provide a consistent set of APIs across different operating systems and platforms. If a Java program is to be truly platform-independent then identical APIs and services must be available on all platforms. The two camps are currently developing different approaches to these APIs, potentially creating a fragmentation.

JavaSoft and a group of other companies (IBM, Oracle, Novell, Netscape etc.) are developing Java Foundation Classes (JFC) that will provide common services on all partner companies' platforms. This group is also strongly criticizing the development and use of platform-specific Java APIs (i.e. ActiveX extensions for Windows) and native method calls. To encourage developers to its approach Sun has proposed a "100% pure Java" initiative to certify Java programs for cross-platform compatibility.

— Future Computing Applications (?) —


Pure Java Class Libraries
Microsoft Extended Class Libraries
or
Pure Java Class Libraries
Microsoft Extended Class Libraries

Meanwhile, Microsoft is actively pursuing its own Java class libraries (Application Foundation Classes [AFC]) and APIs, which integrate tightly with the Windows platform. Microsoft views JavaSoft's approach as the one based on "least common functionality" and considers it inadequate for building advanced applications. Microsoft sees tighter integration with the Windows platform as a means to provide enhanced functionality and programming ease.

The two groups also differ on mixing Java with native objects. For example, the current Microsoft's implementation encapsulates Java objects into ActiveX objects. Such encapsulation results in better integration with platform, but loses some portability.

These different approaches could lead to development of different versions of Java environments, like Unix systems. Some vendors may choose to compromise portability but provide better services by developing Java extensions specific to their platforms. If one set of class libraries becomes dominant, then Java's platform independence and integration will be determined by the underlying design and availability of class libraries on different platforms.

Object Model– JavaBeans

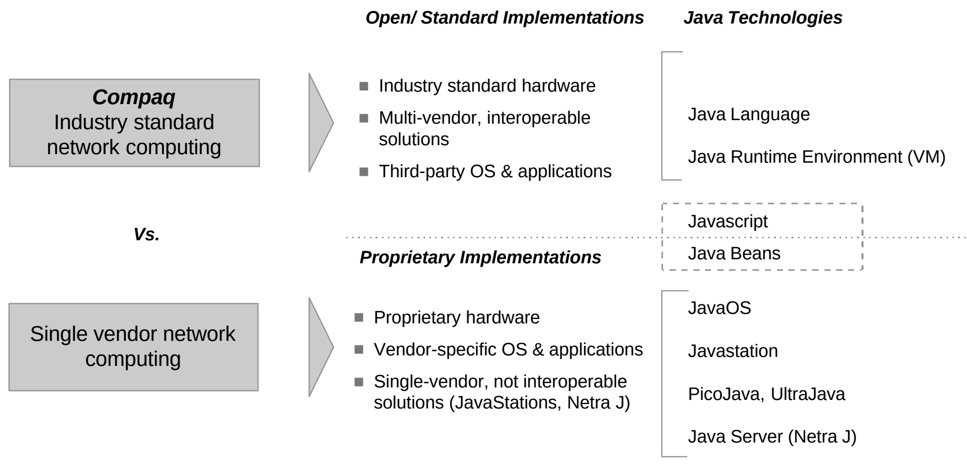
The Java object framework is another area for contention among industry vendors. One of the major opportunities and threats for Java will be its ability to create a workable model for truly distributed, object-oriented computing.

Major issues around technology, complexity, interoperability, performance, and compatibility have limited growth of component-oriented software. For example, CORBA's approach has proven too general to work effectively. Different vendors' implementation of CORBA-compliant objects do not interoperate due to different interpretations of CORBA specifications. Meanwhile, Microsoft's DCOM has worked well for inter-object communications.

JavaSoft has proposed JavaBeans and Remote Method Invocation for communication among Java objects. Vendors are also implementing "bridge" software enabling Java objects to communicate with non-Java objects. However, JavaBeans is yet another model that the industry needs to widely accept before it can become the standard. Today, JavaBeans specifications are controlled by JavaSoft, and not all ISVs have endorsed them.

COMPAQ'S APPROACH

In this confusing environment, Compaq will drive for industry standards, protect its customers' investments, and provide compatible and integrated open solutions.


Open/ Standard Implementations
Java Technologies
Compaq
Industry standard network computing
Industry standard hardware
Multi-vendor, interoperable solutions
Third-party OS & applications
Java Language
Java Runtime Environment (VM)
Javascript
Java Beans
Vs.
Proprietary Implementations
Single vendor network computing
Proprietary hardware
Vendor-specific OS & applications
Single-vendor, not interoperable solutions (JavaStations, Netra J)
JavaOS
Javastation
PicoJava, UltraJava
Java Server (Netra J)

Compaq supports the industry-wide standards of Java as an important object-oriented development language and the Java runtime environment and VM as a critical component of any platform. In this area, Compaq will continue to influence vendors to release their Java technologies to standard bodies.

Compaq views JavaOS as a proprietary OS, very different from the open Java technologies. From Compaq's viewpoint, JavaOS does not match standard OS in functionality, services, and performance. In addition, the current implementations of network computers lock customers into single-vendor solutions. The main focus for customers is reducing TCO while maintaining open environment and support for existing applications. This is best done through a ZAW PC, or a NetPC that integrates Java with Windows while retaining all other benefits of an open platform. Compaq also supports the ZAK initiative that brings these benefits to customers' existing systems.

To protect its customers' investments, Compaq will focus on providing the best Java-based solutions deployed on Compaq platforms. Compaq will provide a high-performance and manageable platform to host the broadest range of Internet/intranet applications.

Finally, Compaq's solutions will not lock its customers into proprietary, single-vendor implementations. Compaq will promote industry-standard network computing that supports the Java language as one of its key components.