2000

Migration from client/server architecture to internet computing architecture

Fauzan Ahmad

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritescholarworks@rit.edu.
Migration from Client/Server Architecture to Internet Computing Architecture

By

Fauzan Ahmad

Thesis submitted in partial fulfillment of the requirements for the Degree of Master of Science in Information Technology

Department of Information Technology
Rochester Institute of Technology

July 14, 2000
I, Fauzan Ahmad, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction must not be for commercial use or profit.

Date: 7/14/2000   Signature of Author: ___________________________
Student Name: Fauzan Ahmad
Student Number: 

Thesis Title: Migration from Client/Server to Internet Computing Architecture

Thesis Committee

Name: Prof. Tim Wells, Chair
Signature: 7/14/2000

Name: Prof. Rayno Niemi
Signature: 7/14/2000
Committee Member

Name: Prof. Edith Lawson
Signature: 7/14/2000
Committee Member
## Table of Contents

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Title</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>List of Figures</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Abstract</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Introduction</td>
<td>8</td>
</tr>
<tr>
<td>3.1</td>
<td>Today’s World of SQL Client/Server</td>
<td>8</td>
</tr>
<tr>
<td>3.2</td>
<td>World Wide Web</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>The Distributed Object Paradigm</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>The Extensibility of Distributed Architecture</td>
<td>13</td>
</tr>
<tr>
<td>4.2</td>
<td>Introduction to Distributed Object</td>
<td>13</td>
</tr>
<tr>
<td>4.3</td>
<td>The Developer’s Perplexity</td>
<td>15</td>
</tr>
<tr>
<td>4.4</td>
<td>The Business Perplexity</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Internet Computing Architecture</td>
<td>17</td>
</tr>
<tr>
<td>5.1</td>
<td>The Open and de Facto Standards</td>
<td>17</td>
</tr>
<tr>
<td>5.2</td>
<td>Introduction to CORBA and IIOP</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Internet Computing Architecture Components</td>
<td>19</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction to Cartridge</td>
<td>20</td>
</tr>
<tr>
<td>6.2</td>
<td>Inter-Cartridge exchange (ICX)</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>Extensible Database Servers, App. Servers, Clients</td>
<td>24</td>
</tr>
<tr>
<td>7.1</td>
<td>Family of Clients in Internet Computing</td>
<td>24</td>
</tr>
<tr>
<td>7.2</td>
<td>Introduction to ActiveX/COM/DCOM?</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>Use of Universal Application Server</td>
<td>26</td>
</tr>
<tr>
<td>8.1</td>
<td>Internet Computing: A Scaleable Application Server</td>
<td>27</td>
</tr>
<tr>
<td>8.2</td>
<td>The Universal Server</td>
<td>27</td>
</tr>
</tbody>
</table>
Applications based on Cartridges

Development of Applications based on Cartridges

Management of Applications based on Cartridges

Summary

Appendix A

Appendix B (Case Study)

References
1. LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Today's World of SQL Client/Server</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>World Wide Web</td>
<td>11</td>
</tr>
<tr>
<td>4.1</td>
<td>The Distributed Paradigm</td>
<td>14</td>
</tr>
<tr>
<td>6.1</td>
<td>Internet Computing Architecture</td>
<td>19</td>
</tr>
<tr>
<td>8.1</td>
<td>Universal Application Server</td>
<td>26</td>
</tr>
<tr>
<td>S.1</td>
<td>IT Environment in Today's World</td>
<td>32</td>
</tr>
<tr>
<td>C.1</td>
<td>ABC Internet Computing Architecture Model</td>
<td>34</td>
</tr>
<tr>
<td>C.2</td>
<td>Workspace Architecture</td>
<td>36</td>
</tr>
<tr>
<td>C.3</td>
<td>Persistent Object Service</td>
<td>42</td>
</tr>
<tr>
<td>C.4</td>
<td>ABC Three-tier Architecture</td>
<td>43</td>
</tr>
<tr>
<td>C.5</td>
<td>Thin Client Architecture</td>
<td>46</td>
</tr>
</tbody>
</table>
The Internet Computing Architecture helps in providing an object-based infrastructure that can be used by the application developers to design, develop, and deploy the n-tiered enterprise applications and services. For years of distributed application development, the Internet Computing Architecture has helped in providing various techniques and infrastructure software for the successful deployment of various systems, and established a foundation for the promotion of re-use and component oriented development. Object-oriented analysis is at the beginning of this architecture, which is carried through deploying and managing of finished systems. This architecture is multi-platform, multi-lingual, standards-based, and open that offers unparalleled integration capability. And for the development of mission critical systems in record time it has allowed for the reuse of the infrastructure components.

This paper provides a detailed overview of the Internet Computing Architecture and the way it is applied to designing systems which can range from simple two-tier applications to n-tier Web/Object enterprise systems. Even for the best software developers and managers it is very hard to sort through alternative solutions in today's business application development challenges. The problems with the potential solutions were not that complex now that the web has provided the medium for large-scale distributed computing. To implement an infrastructure for the support of applications architecture and to foster the component-oriented development and reuse is an extraordinary challenge. Further, to scale the needs of large enterprises and the Web/Internet the advancement in the multi-tiered middleware software have made the development of object-oriented systems more difficult.

The Internet Computing Architecture defines a scaleable architecture, which can provide the necessary software components, which forms the basis of the solid middleware foundation and can address the different application types. For the software development process to be component-oriented the design and development methodologies are interwoven. The biggest advantage of the Internet Computing Architecture is that developers can design object application servers that can simultaneously support two- and three-tier
Client/Server and Object/Web applications. This kind of flexibility allows different business objects to be reused by a large number of applications that not only supports a wide range of application architectures but also offers the flexibility in infrastructure for the integration of data sources. The server-based business objects are managed by runtime services with full support for application to be partitioned in a transactional-secure distributed environment. So for the environments that a supports high transaction volumes and a large number of users this offers a high scaleable solution.

The integration of the distributed object technology with protocols of the World Wide Web is Internet Computing Architecture. Alternate means of communication between a browser on client machine and server machines are provided by various web protocols such as Hypertext Transfer Protocol and Internet Inter-ORB Protocol [IIOP]. Protocols like TCP/IP also provides the addressing protocols and packet-oriented transport for the Internet and Intranet communications.

The recent advancements in the field of networking and worldwide web technology has promoted a new network-centric computing structure. World Wide Web evolves the global economy infrastructure both on the public and corporate Internet's. The competition is growing between technologies to provide the infrastructure for distributed large-scale applications. These technologies emerge from academia, standard activities and individual vendors.

Internet Computing Architecture is a comprehensive, open, Network-based architecture that provides extensibility for the design of distributed environments. Internet Computing Architecture also provides a clear understanding to integrate client/server computing with distributed object architectures and the Internet. This technology also creates the opportunity for a new emerging class of extremely powerful operational, collaboration, decision support, and e-commerce solutions which will catalyze the growth of a new networked economy based on intra-business, business -to-business (B2B) and business-to-consumer (B2C) electronic transactions. These network solutions would be able to incorporate legacy
mainframe systems, emerging applications as well as existing client/server environment, where still most of the world's mission-critical applications run.

Internet Computing Architecture is the industry's only cross-platform infrastructure to develop and deploy network-centric, object-based, end-to-end applications across the network. Open and de facto standards are at the core of the Internet computing architecture such as: Hyper Text Transfer Protocol (HTTP)/ Hyper Text Markup Language (HTML)/ Extensible Markup Language (XML) and Common Object Request Broker Architecture (CORBA). It has recognition, as the industry's most advanced and practical technology solution for the implementation of a distributed object environment, including Interface Definition Language (IDL) for language-neutral interfaces and Internet Inter Operability (IIOP) for object interoperability. Programming languages such as JAVA provides programmable, extensible and portable solutions throughout the Internet Computing Architecture. Internet Computing Architecture not only provides support, but also enhances ActiveX/Component Object Model (COM) clients through open COM/CORBA interoperability specifications. For distributed object-programming Java has also emerged as the de facto standard within the Internet/Intranet arena, making Java ideally suited to the distributed object nature of the Internet Computing Architecture. The portability that it offers across multi-tiers and platforms support open standards and makes it an excellent choice for cartridge development across all tiers.
3. INTRODUCTION

Over the years, innovations in computer software/hardware and network technology have continuously increased performance and reliability while reducing cost. These innovations in technology have allowed multiple generations to evolve: mainframes, client/server, Internet/Web, and distributed objects.

3.1 Today’s World of SQL Client/Server

The networked economy is changing the way that corporations are developing and deploying applications. It is the intersection of public and private networking technologies with 24-hour global e-commerce. In the 80s, we had the industry Personal Computers with graphical user interfaces (clients), high-end servers to house major databases, and to connect these two we had the Ethernet LANs. Companies gained peace of mind as they rebuilt, rightsized, and downsized their mainframe-based computing infrastructures with the new industry standards being emerged on the client/server.

After fifteen years, with the introduction of the Personal Computers (PC’s), client/server became the dominant architecture with the base for the new upcoming enterprise computing systems. However, a few of the legacy applications still reside on the mainframe technology. Today’s client and server world not only runs mission critical applications natively but also helps in providing robust bridges for their legacy applications and comprised data.

Client/server world has created a few challenges even though PC clients have always been a cost-effective option to proprietary systems and to character-mode terminals. As the clients became “fat” (the FAT file system is named for its method of organization—the file allocation table. The FAT file system is a simple file system originally designed for small disks and simple directory structures. Its design has been improved over the years to work more effectively with larger disks and more powerful personal computers) the demand increased for more software and hardware, which made the client administration increasingly difficult. Significant
resources are required for the management of multiple software’s packages with several versions on different clients. In extending applications the client software once developed offers limited reusability.

Figure 3.1[10.a], Today’s World of SQL Client/Server

With the advancement in Internet/web technology, there is a steady decrease in the cost of high-end servers, and increase in the total cost of ownership of the Personal Computer clients, which are prompting Independent Software Developers (ISD) to extend their strategies to client/server to embrace the Internet/Web.

The ultimate goal must be to create a applications development platform that is cheaper to develop and deploy. Moreover, developers are designing a Internet model based on a browser based thin client (The thin-client/server computing model involves connecting thin-client software or a thin-client hardware device with the server side using a highly efficient network protocol such as Citrix’s ICA. The thin-client/server architecture enables 100 percent server-based processing,
management, deployment, and support for mission-critical, productivity, Web-based, or other custom applications across any type of connection to any type of client hardware, regardless of platform. The client hardware can include Windows-based terminals, PCs, NetPCs, network computers, Apple Macintosh computers, or UNIX devices), the application logic based middle tier, and a database tier which can hold the company's business rules and data as in the client/server model.
3.2 World Wide Web

The emergence of the Internet/Web has created another wave of network technology that is anchored by low-cost, simple, easy-to-deploy web technology with high bandwidth networks. The explosion of the World Wide Web was driven by documents - Internet pages that contain rich graphics and text, and were interlinked in powerful ways. The Web technology was initially deployed with these interlinked documents housed on servers that used a centralized approach - the corporate servers’ “glass house” picture accessed by millions of browser.

This Internet/Web technology is the first true large-scale, multi-vendor, open, distributed application of Internet computing and has provided a very powerful and very successful platform to handle documents by browsing and surfing. Nowadays, more and more functionality is being added to the public Internet and the corporate Intranet to enhance its performance and to make it easier for the vendors to create, share, search and deal with the day-to-day documents.

Figure 3.2 [10.b], World Wide Web
The Internet/Web solves many of the challenges faced with the client/server computing like the fat client problem. The Web on the other hand provides a low-cost, and easy to use, open deployment platform.

Still the Web faces more challenges as it evolves from a document library to an e-business platform to conduct secure transactions on corporate Intranet and public Internet's. But as a universal platform for large-scale distributed applications it still lacks in security, scaleability, mission-critical services, and a simple means of creating, managing, debugging, deploying, and operating applications. It must:

- Enhance the integrity, data security, and distributed transaction support
- Enhance the reusable extensibility, and provide robust, scaleable architecture

These kind of challenges magnify the need for Internet-savvy programs/objects that can offer work on multiple operating systems, languages, tools, hardware, applications, and can also offer robust extensibility.
4. THE DISTRIBUTED OBJECT PARADIGM

4.1 The Extensibility of Distributed Architecture

The next generation approach for application development and deployment is considered to be the Object Oriented Technology. It provides the mechanism for more efficient programming and quick responses to business needs and opportunities. To develop the pieces of an implementation, at variable times, by different project teams and Independent Software Vendors, the Objects also allow a “pen” environment.

4.2 Introduction to distributed object

An Object is considered to be a self-contained package comprising both of code and data. Since an object is limited only to a number of well-defined methods and to the data it encapsulates, its design must contribute to product quality and safety. With the principle of inheritance, the use of object speeds up the development time and cuts down the cost and time-to-market as the capabilities of an object may be extended and embraced to build another different object. In a system, the objects may differ but might have a response for the same point in their own way just like polymorphism, which makes them easy to manage. The distributed objects can live anywhere on the Internet with the assistance of the Object Request Broker (The Object Request Broker (ORB) is just a set of CORBA specifications. It refers to a collection of software libraries used either by CORBA clients to access distributed services or by CORBA servers to make their services distributed).

In the Information Systems world, the Object Oriented Technology enhances the productivity in programming by allowing the software developers to develop software as a more useful and manageable piece. In the network-centric environment, objects behave as distributed objects, which can operate on different operating systems, languages, and network. Same as the client/server approach, the computing and network resources are permitted by distributed objects to be applied effectively across the network. For example, Implementation of an application as a set of objects
with some running on a corporate server and others running on end-user PCs or a departmental server.

![Diagram of the Distributed Paradigm](image)

**Figure 4.1[10.c], The Distributed Paradigm**

The software developers can create new applications by connecting software objects that contain the enterprise business logic. The features of the objects can be organized, shared and evolved in parallel by the team of developers. Providing interoperability and platform independence the distribution of software objects can be done across the network.
The partition of distributed objects can be done across the network for better utilization of system resources, which was previously impossible with the client ‘fat’ architecture. With existing legacy applications and interfaces the distributed objects can co-exist through object wrappers.

With the logical progress of network technology and its migration from the client/server world to network-centric object computing, the technology customer’s need has arrived. Thus, creating a dilemma for developers and business alike.

4.3 The Developer’s Perplexity

With the advancement in technology and the advent of new system environments, the choice in the number of tools and technologies has increased dramatically for the software developers. The competition is growing between the technologies for the role to provide the infrastructure for distributed large-scale environments - the underpinnings of the newly emerged networked economy. These technologies emerge from academia, standard activities and individual vendors. Every technology, offers unique possibility for the enhancement of development productivity or for providing a new capability to the user, that includes Java, HTTP/HTML, CORBA, ActiveX, IIOP, DCOM, JavaScript’s, and so on. As a developer makes a commitment to a tool or technology it has the capability to limit the functionality for re-targeting or for adapting the core capabilities of the system that limits the ability to respond to any change.

A significant amount of investment is done in software, training, and operational infrastructure when a developer chooses a distribution technology, a tool or API. And investment in these adds a significant risk. With relatively little compromise, the developer ideally leverages the best of each technological environment. For example, a SQL client/server application having a web front end or Web applications capable of performing database transactions can be written by a developer.
4.4 The Business Perplexity

The growth and the rapid adoption of the Internet/Web technology by both the public Internet and the corporate Intranet have made a huge progression from just a mere stage of browsing the documents to a real demand for business applications. Implementation using various extensions of the core technology is already in progress. Enhancements are being made into these applications to provide messaging, robust transaction, and data access services, which is typical of client/server applications. As the businesses not only want to deploy the robust Internet Applications but also want to protect their significant investments that had already been made in client/server technology and its integration with the legacy system.

Integration of these legacy systems with the new Internet and Object Oriented Technology, creation of operational, Intranet and e-commerce solutions that can run their business, is the real business challenge.
5. INTERNET COMPUTING ARCHITECTURE

The business requirements are met by the combination of the new and existing technologies, these different technologies evolve as the industry moves these into the production usage, and that no need remains for another proprietary, or another vendor driven solution (see Appendix A). Open and de facto standards: CORBA and HTTP/HTML are at the core of the Internet Computing Architecture.

5.1 The Open and de Facto Standards

Developers and Business are both insulated from the risks associated with the growth and newly evolving technologies by providing a unified, standards based architecture for the web, client/server, and the distributed objects. Open and de facto standards support can guarantee that developers can use a language which suits the application, their client of choice, and for rapid deployment an appropriate programming model.

5.2 Introduction to CORBA and IIOP

CORBA is mainly a specification of a consortium known as the Object Management Group (OMG) which includes approximately 600 companies to represent the entire range of the computing industry. It stands for the Common Object Request Broker Architecture. Distributed Architecture is defined by CORBA with an open software bus through which objects from multiple vendors can inter-operate while using different operating systems. Defined atop TCP/IP which is the necessary communications protocol, object interoperability is ensured by the Web/Internet Inter-ORB Protocol.
Internet Computing Architecture is a combination of the CORBA’s object oriented technology with the web/Internet technologies of HTML and HTTP which forms the basis for the distributed computing architecture in the networked economy. For implementation of a distributed object environment including IDL for language-neutral interface and IIOP for object interoperability, CORBA has been recognized as the industry’s most advance and practical approach technology.

Internet Computing Architecture not only supports but also helps in enhancing ActiveX/COM clients by interoperability specifications of open COM/CORBA as ratified by the Object Management Group and the additional services provided by various ICA vendors. To deliver real business solutions by developers, this approach allows for the mixing and matching of components to create applications. They can create corporate network and Web applications that combine:

- The robust client/server world,
- The object-oriented world’s component capabilities,
- The flexibility of the Web, low deployment cost, and the ease of use.

Existing client/server application with Internet Computing Architecture can take the advantage of Web/Internet technology with a minimal change and integration of new Web/Internet applications can be seamlessly done with existing client/server system without a complete IS upheaval.
6. INTERNET COMPUTING ARCHITECTURE COMPONENTS

Internet Computing Architecture's key architectural components are:

- Cartridges - “Pluggable” objects that are easy to manage and capable of providing extensible functionality
- Standardized interfaces and open protocols - that are a means of communication among cartridges through a software bus known as Inter-Cartridge Exchange (ICX)
- Application Servers, Database Servers and the Extensible Clients

⇒ Any Client
⇒ Universal Application Server
⇒ Universal Server
⇒ Development Environment and management of Cartridges.

*Figure 6.1[8.a], Internet Computing Architecture*
6.1 Introduction to Cartridge

It is component-based software. Simply put, a cartridge is a program component. To use a cartridge, one must install and register it. This means an application cartridge is installed on the application server and registered. Cartridge uses an IDL (Interface Definition Language) which is a language-neutral interface that allows identifying the cartridge to other objects in a distributed environment. Cartridges can be built with PL/SQL, SQL, JAVA, C/C++, Visual Basic, etc.

These Cartridges have access to Universal Cartridge Services:

- Installation – this is the mechanism to get the program code
- Registration – this is the mechanism to bring a particular cartridge into service
- Installation – this is the mechanism to activate a particular cartridge to handle the demand for its services; its advantage is that it can be on several distinct machines, and as demand rises more instances being activated and destroyed as demand drops
- Invocation – this is the mechanism for providing a task to a cartridge by an application or another cartridge for performing one of its service
- Administration – this is the mechanism to control updates, versions, etc.
- Monitoring – this is the mechanism for operations and overseeing a cartridge's activity
- Security – this is the mechanism to assure the appropriate use of computing resources and protection of data

A few of these services are a part of CORBA standard. Cartridges also help make CORBA objects manageable and deployable. For example, a browser plug-in based on CORBA, such as JAVA applet-using IIOP to server evolves into a cartridge after it access Universal Cartridge Services. With CORBA and IIOP all of this can be accomplished.
6.2 Object Bus: Inter-Cartridge exchange (ICX)

Communication of cartridges with each other across a distributed network is enabled through an object bus known as Inter-Cartridge exchange. For this communication, it uses both IIOP and HTTP protocols to make translations that are necessary to cross between distinct environments.

A whole set of libraries and services are implemented for ICX that resides within the various computers attached to the network. With these libraries, cartridges can access:

- Clients
- Services
- Servers
- Other cartridges

The ability to create an interface between the proprietary environments and the Internet Computing Architecture is also provided by ICX.

Integration of ICX with COM

Communication of ActiveX/COM clients with cartridges can be done via a bridge. ActiveX/COM clients that support Java have access to cartridges, as well as the messaging, scaleable transaction, and queuing services.

Integration of ICX with Java

Internet Computing Architecture provides the Java language integration with IIOP/CORBA through Java mapping for CORBA IDL, JDBC for the access of open database, and client or server side Java language over CORBA/IIOP
Integration of ICX with Legacy Systems

Integration of mainframe is provided through encapsulation of the legacy system interface within cartridges and interoperable services for queuing, messaging, and transactions.

Additional Cartridge Services

With the above structure, HTTP and/or IIOP can be used as the base to write distributed applications. But to build “real” distributed applications, HTTP and IIOP does not provide the basic communication facilities. Other facilities, that includes security services, persistence, transactions, naming and directory services, concurrency control, and many others, also play pivotal roles.

For the programming environment to be more productive, new functionality has been added by additional cartridge services to add to the capability, which can be used by the application in place of coding into it. These kinds of services are based on CORBA standards so as to ensure openness.

Scaleable Cartridge Services can be used by cartridges, such as:

- Transactions - this is the capability to bracket a series of functions and to treat them as an atomic operation
- Messaging and queuing - this allows asynchronous interaction between cartridges
- Data access - this is the ability to access database server services through open, published interfaces
In addition, depending on where the cartridge expects to be hosted. A cartridge has a privilege to access the Specialized Cartridge Services:

- Access to extensible database services for data cartridges in the database server
- Access to scaleable load management and distribution services for Application server cartridges in the application server
- Access to standard user interface services for Client cartridges
7. EXTENSIBLE DATABASE SERVERS, APPLICATION SERVERS, AND CLIENTS

Heterogeneous, cross-platform solutions are possible by fully extensible database servers, application servers, and clients. Throughout Internet Computing Architecture, the cartridge technology provides extensibility.

7.1 Family of Clients in Internet Computing

There are several competing standards for interoperability on the client tier.

Netscape’s ONE client since it uses IIOP it is likely to be a browser based client capable of plugging directly into the architecture. For reusable user-interface components, On the Windows desktop, ActiveX standard by Microsoft has fully feathered into an important technology. With the proliferation of Web based Intranet/Internet applications, both HTML and Java on the client has emerged as the open and de facto standards. Last, but not least, Network Computer has generated a great deal of interest as a low-cost, manageable alternative to the client PC in a network-centric architecture.

7.2 Introduction to ActiveX/COM/DCOM

For Microsoft’s Object Model, ActiveX controls are well-defined component packages, which offer object-oriented extensibility on Windows 95/98 or NT machines, typically the client desktops. For object interoperability across networks and between machines, Microsoft offers DCOM (Distributed COM).

Homogeneous-system interoperability mechanism is represented by a single-vendor “standard,” DCOM, in sharp contrast with CORBA/IIOP, which is a multi-vendor standard for networks comprising heterogeneous systems.
It is significant to use secure industry standards when building with software components which makes it positive that if a particular part fails, it can easily get replaced. Also it makes it easy to replace a existing part when upgrading. Planning can be done ahead of time, when using industry-standard approaches to component-building as a huge selection of pre-built components is provided assuming that they will be working together when trying to combine them.

The advantage of Internet Computing Architecture is that it provides developers a choice in C/C++, Java, JavaScript, Visual Basic, and SQL-based languages to create their own cartridges and also can support any of these clients. As by using ICX the clients and cartridges can communicate and the development centers or organizations no longer requires to make a singular decision based on environments in which they require to develop. Each component cartridge guarantees to interoperate across the architecture standard protocols.
8. USE OF UNIVERSAL APPLICATION SERVER

Using Internet Computing Architecture, the Universal Application Server has an important role from the applications point of view. It can take the responsibility for application coding that might otherwise reside at the client or on the database server thus it can act as a platform or stage for reusable business logic. The clients using Internet Computing Architecture thus become more lightweight and manageable and application developers become more specialize: interface designers can create the client, while domain experts can code the business rules.

Figure 8.1[^10.d], Universal Application Server

For the Web Request Broker, a Scaleable Application Server is used to support the application cartridges based on HTTP and HTML programs by which the Web cartridges gets the services of the CORBA compliant ORB which provides the integration of object-based applications and Web-based applications.
With the addition of scaleable transaction services and HTTP, ICA includes a robust application server for the Web/Internet.

8.1 Internet Computing: A Scaleable Application Server

The Web Request Broker is evolved into ICA's Universal Application Server based on an Object Request Broker and Scaleable Cartridge services and using IIOP interoperable and connectivity services. Through ICX over IIOP or HTTP these kinds of services would be available to all cartridges.

Universal Application Server will also incorporate other connectivity protocols. The interactions and interfacing with existing client/server mechanisms is handled by the Connectivity Broker, such as the installation using SQL*Net RPC or special-purpose communications environments, such as the wireless services.

8.2 The Universal Server

The Universal Server provides robust, scaleable data storage and manipulation. In addition to its traditional relational data support it can now manage more current types of data, which includes audio, video, text, and spatial data that provides powerful new ways for managing, manipulating, and delivering of data in the networked economy.

Significant performance and management advantage can be provided in many applications by applying a little logic to the stored database data. Application developers, with stored procedures can write the database intensive code - such as on sales data computing aggregate statistics - to leverage powerful database features on the server like parallel query execution. By the use of object-relational database technology ISV's (Independent Software Vendor's) are capable of extending the database engine core capabilities by creating new sophisticated functionality data types, like pattern matching on audio or image data.
To extend the database Safely within Universal Server, the extensible database services provides administrative tasks, indexing, query engine and similar key components of the database kernel. Developers can also create specialized data cartridges for the implementation of efficient new data types by the encapsulation of extensible database services through IDL and SQL in a high level interface.

Data cartridges, developed using standard tools are managed inside the database by Universal Cartridge Services which provides a safe, standards-compliant technology for the delivery of new functionality in the database.

Universal Server cartridges provides a standard environment for the implementation of data logic using SQL, IDL, and Java. Unlike other forms of extensible database cartridges are also networks enabled with Universal Cartridge Services standard set which also includes the installation and registration. For the support of CORBA, HTTP and IIOP, Inter Cartridge Exchange extends Universal Server, enabling the direct two-way communication between the cartridges in the client, the application server, and the database server.

And as of today the data and services in the Universal Server will continue to be accessed via SQL*Net.
9. APPLICATIONS BASED ON CARTRIDGES

9.1 Development of Applications based on Cartridges

A new generation of software tools is required for building the applications that requires a distributed computing network of servers, application servers and clients. Using objects makes it easier to build applications for a potentially large computer network as easy as it is to build application for a one computer.

With Internet Computing Architecture and cartridges, applications can be created with multiple development environments, tools, and techniques. Language - centric and Object Centric are the two classes of development environments.

In the Language - Centric environment the dominant influence is the programming language while the tools focus on using that language to create code. Choice among the tools is available for many current languages from C, C++ to CORBA to Java and cartridges can be created by using any of these languages.

Some of the development tools for the Web/Internet client/server includes:

- Developer/2000 : For the development of Web-based SQL forms
- Designer/2000 : For the design of Web-based SQL CASE
- Any JAVA, Enterprise Java Beans (EJB) development tool
- Any C, C++ or Visual Basic development tool

In the Object-Centric environment, the language is secondary. The components (cartridges) have the primary focus. The mechanisms for the design tasks is provided by the tools, such as to decide what type of functionality is performed in which place - universal servers, application servers, or clients.
An example for this type of approach is Sedona, which provides an object-oriented deployment, analysis, and design architecture for deploying and assembling distributed object applications on top of Internet Computing Architecture. Sedona’s key functionality includes:

- Object modeling and the Cartridge assembly
- Design and the Development of components and object-based applications with object-oriented programming tools.
- Team development based on repository

### 9.2 Management of Applications based on Cartridges

ICA vendors such as Oracle provide tools such as Enterprise Manager for the management of complete Oracle environment. A lightweight layer of the most common services is provided by the console agent architecture for many management applications, which includes installation, configuration, administration, and monitoring. The various components of the Oracle Environment is managed by the enterprise manager which includes database, systems, applications, networks and can be extended for the management of cartridges on all tiers of Internet Computing Architecture.

A family of web based tools is also provided by the Enterprise Manager, by which a standard web browser can be used for database management, monitoring, and administration. This web-based tool is incorporated in the web request broker as a cartridge, with a functionality of automatic discovery mechanism for other web based management tools. To enhance the extensibility, the APIs are provided for other cartridge-specific management tools to hook to the Enterprise Manager console.
10. SUMMARY

The direction of Internet Computing Architecture is for the design, deployment, and management of today’s distributed mission-critical enterprise systems. It provides the combination of the robust world of client/server, the easiness of use and deployment of Web-based environment with the extensibility of the object-oriented world. Internet Computing Architecture provides a solution for the concerns and needs of multiple requirements of multi-audiences.

For in-house and independent developers, Internet Computing Architecture represents a framework for the integration of new and pre-existing software components with a choice of best-in-class technologies and environments for their implementation. With its open architecture the developers can create reliable, serviceable, accessible, and easily extensible solutions.

For the info-tech managers, Internet Computing Architecture ensures simplified management and smooth deployment of powerful new applications. For business managers, Internet Computing Architecture represents a cost-effective path to implement new object technologies by managing the minimization of development tasks and leveraging the existing hardware and software investment while going forward.

For CEOs, Internet Computing Architecture helps in providing the peace-of-mind of a simple, safe and globally supported solution, which evolves from proven technology and incorporates the most widely adopted multi-vendor open standards. Internet Computing Architecture helps to provide a bridge from today’s solution to create a technical environment in which the distributed network applications can be easily build, deployed, and can be managed, by using objects from multi-vendors which can be accomplished by combining:

- object world - extensibility
- client/server world - robustness
- Web/Internet - the ease of use and deployment
In real-world systems, these are the three principles at the core of technical platforms. Internet Computing Architecture combines these fundamental assets to bear on the problem of network usage for real applications - so as something can be done in the networked economy. To “win” and dominate the landscape no one technical community is required. Yet it does not create another technology.

Thus, it does combine these three technologies, which has already achieved the industry’s acceptance and proved success, in the field of creating a future platform and a roadmap showing how to get there. Unified framework which shows the diversified worlds of distributed computing, the Web/Internet, and the client/server to have a common network computing model that is based on existing standard architecture.
APPENDIX A

Industry Standards

Accepted vs. Competing standards

Currently, Structured Query Language (SQL) has been widely accepted as the industry standard database language. In several other software areas, there are multiple competing “standards” instead of just one. As considering the desktops (PCs), multiple competing standards” are there in the field of graphical user interfaces, operating systems, network protocols, including the web browser. Several emerging competing standards are at the mid-tier application servers, and all of them demand custom development but none of them provide the administrative tools for the application server. The main dilemma is to choose which one? What happens if the “wrong one” is selected?

Internet Computing Architecture vendors, in order to minimize the effects of change have made their applications, servers, and tools portable which support all operating systems, computers, object models, user interfaces, and networks.

With the advantage of portability, the organizations have the confidence to adopt and work with one set of standards, and then plan their migration strategy as the new standards emerge. Support for different operating systems and protocols is provided by ICA vendors such as MS-DOS, Windows, UNIX, TCP/IP, and as well as ActiveX and CORBA clients.
APPENDIX B

Company Name: ABC
Case Study: Migration of ABC from client-server to Internet Computing Architecture

The Internet Computing Architecture is used to make a complete solution and to develop applications that promote the use of Java or any technology, which is related to it either entirely or in the form of applets. These programs based on Java use technologies like thin-client interface and CORBA to gain access to business objects, transactions, and shared services. Deployment of these types of applications can be done within the Intranet by using the technologies of Universal Client.

![Diagram of Internet Computing Architecture]

Figure C.1[13], ABC Internet Computing Architecture

To develop a reliable, scaleable application, the back-end architecture to support this model must support load balancing, transactions, fault-tolerance, and session management. Fundamental services are provided by ABC's Internet Computing Architecture and permit the deployment of the business objects within this model.
Principal Components:

ABC’s Internet Computing Architecture comprises of a number of Object services (a package that implements a key component of the overall object management system). There are numerous components threading throughout this computer architecture.

Among these items some are conceptual and control the object services design, which utilizes these items. While the others are more tangible objects existing within software.

Object Workspace:

A unique architecture prevails at the heart of ABC for the management of objects known as Object Workspace, which provides for multiple data sources, garbage collection, caching, and distributed transactions. Basically a workspace is a manager of all objects, known as an n-memory container, and of the relationships that are being fetched or created during any transaction. The advantage of its being linked to transactions is that it eliminates the need for writing a clean up and complex undo code and ensures the release of all resources and objects after the complete of the transaction. For distributed transaction management and application partitioning the fundamental component provided by Workspace model.

A central function of this workspace is an object cache, which keeps tracks of all transactions to ensure that a particular object is only read once. Object cache helps in the management of object references. As a result, it improves the performance and helps the programmer so he doesn’t need to keep track of the objects and the changes made to them. All object changes are flushed to the database with the commit of a transaction in the application.
The diagram below illustrates the Workspace Architecture:

![Diagram of Workspace Architecture]

**Figure C.2**[12.b], *Workspace Architecture*

A workspace is created with a start of a transaction and the application needs to specify the database in a two-tier architecture to connect the workspace. For the transaction to start in a distributed environment the client needs to name the initial application package they wish to use. Dynamically, this name gets mapped to the server by using the Distributed Service Coordinator and creates the workspace both on the client, which initiated the transaction and on the selected server.

**Business Object**

A business object is a software representation of real world entity, concept or a process. The entity, in several architectures, is assumed to have a specified interface, which allows these objects to accept events, participate in rule checking, and be persistent. All of these events and services are available in ABC’s architecture without any need of object hierarchy being imposed, which implies that there is no requirement of base classes. These are the principal components in the transaction creation.
Business Component

A view or one or more business object (a business object reflects a particular business process, concept, or entity) is known as a business component, which is commonly used in a three-tier architecture. They help to eliminate the problem of reading large objects to access only a few fields. For application to have its own business components tailored to the application these classes could be defined independently. These are generated as Java Bean classes in Java.

Package

A package is a collection of procedures and functions bundled together usually by virtue of similar functionality. All internal procedures and functions are recorded in the data dictionary as one singular stored package. It forms a significant object as a set of business classes and has its basis in Universal Modeling Language (UML). Below is the diagram showing implementation of insurance policy management system representing three packages.

Policy Package is the primary package for implementing the policy model and for computing quotations and gathering information. A significant number of advantages are offered by the package architecture. Partitioning and management of heavy business models is allowed as a whole entity. It represents a good management of Object service from ABCs point of view that has instances across the servers.

By supplying just the name, a transaction can be started by the application within a particular package. From the initial design phase to the final stages of server deployment these packages are conceptually very important.

Object Service

It is a facility or a package for the implementation of a key business component of the object management system.
DESIGN ARCHITECTURE

ABC’s Architecture offers different tools and techniques to generate source code of object model, foster reuse, and rapid prototype development to aid in the object-oriented design and analysis which illustrates ABC’s design process and how the tools are used to aid in the analysis and design of software.

Scope, Objectives, Approach and Quality Plan

A package of support, education, and services is developed to ensure the success of the project, within a defined framework of quality, time, and cost. The initial statement of this framework is defined as the Scope, Objectives, and Approach which, together with the companion Quality Plan, enable ABC to work with clients to deliver business solutions based on Internet Computing Architecture.

The purpose of the Scope, Objectives, and Approach document is to provide a reference to ABC for resources working on the project at any time and for any duration. This document also is the source for implementation approaches, direction, risks, benefits, and assumptions.

With the extensive Object Oriented Technology experience, ABC has a standard process for the design and construction of software systems that leverages their tools and architecture.
Following are the four steps that it follows based on the object methodology:

1. Analysis
2. Design
3. Implementation
4. Deployment
Analysis

It is a phase to understand the system that we are to build and its actors, components, requirements and workflow. Not only this but we have to understand the

- Project scope: objectives, milestones, and critical success factors
- Project approaches: management, environment, and other business initiatives
- Strategies for technical processes for the method and the approach defined
- Project policies, risks, and assumptions
- Acceptance criteria and change management

In addition, attempts are made to understand the project scope and to provide the clients with the early estimates on time and cost.

For the analysis of the project the following tasks should be conducted:

- Think and Document the problem: May be two or three lines. This documents the project’s purpose
- Interview the clients and the experts and start gathering the requirements
- Start defining the actors who would be involved
- Start making a capability maturity model. The final project would have a entirely different shape than the initial shape
- If the requirements is to use the existing database, add/correct the relationship and import the table as classes
- Document rules for business
- Make one or more prototypes. Approach would be utilize the previous object model after generating a set of business classes and then start making GUI on this.
- Make a preliminary estimate of the project in terms of time and cost
Design

Object-oriented design process is mainly about to define details. During analysis, an overview of the whole project is done and a good picture of the system is projected with the types of requirements needed in detail. Various errors, omissions, etc. would be discovered during this design phase which will prompt the developer to go back to the analysis phase and then come back to the design phase. This process is normal with any project and is a iterative process.

After the completion of the design and Analysis a completed formal draft of the document is made.

Various tasks are included in the design phase generally they are:

- Thorough investigation of the existing design pattern and import design of a appropriate model
- Creation of diagrams, workflow, etc.
- To establish a relationship between the classes so as to allow the objects in the diagrams to communicate to each other.
- Work with the models developed during the analysis phase and develop various scenario and diagrams so as to model the processes behavior
- Utilization of these scenarios for the transaction boundaries development for Workspace
- Utilization of data flow diagrams specially for non - technical people for whom its a excellent communication tool
- To establish architectural foundation so as to describes the system’s basic structure. For the success of an object oriented project, it is very essential to establish a Architectural foundation
- To develop the business objects packaging from the architecture. To design the business component for thin-client
• Mapping of the object model to database for the storage of the business objects. Tuning of the storage model so as to include the design of a physical database
• To create various scripts for the user interface. User scenarios and various screen shots are mainly developed during the design phase. Sometimes it is helpful in saving a incredible amount of time at the end
• To develop a plan for testing of these scenarios
• To develop a good detailed logical project plan. Many time and cost factors rely on a good detailed project
• To create a documentation package for the diagrams and documents. This package with all the specifications helps developers in building a good system

Get the acceptance of the design with all the team members. Use of the tested code generation tools, next step in implementation to create database schema, object models, business components, and CORBA classes.

After the start of the implementation phase, the developers start applying the business logic and start the conversions and interface process.

**Software Development Components**

The Object Schema Compiler is used to generate and compile the source code of the components, interface classes, and business objects at that point the line is crossed between analysis, design, and implementation. Which is quite often when using object development method. Code generated by the Object Schema Compiler is written to the components of the ABC management and the container classes and data types that are defined in the base set library are utilized.

A broad overview of each of the ABCs major frameworks and services is described below. This is the ABC Architecture from the viewpoint of the business and application development.
Persistent Object Manager: Two-tier

Figure C.3[14], Persistent Object Service

The foundation technology of the ABC Structure is contained mostly in the commercial offering of ABC framework, which is mainly geared towards the client/server two-tier architecture. Generally the main focus of this architecture is client-side object management and object persistence. These objects exist in the three-tier architecture with a little change from two-tier architecture.

As the main focus of this paper is on three-tier and Internet Computing Architecture we will avoid elaborating on two-tier architecture. While many features of two-tier architecture are described throughout this thesis. However, it is crucial to specify that the two-tier architecture would be supported as an integral part of this three-tier architecture.
An application can be implemented in a two-tier which scales from a single user, by using personal database, to an application that runs in a multi-tier, multi-user, and mission critical environment and even without any change in the source code.

Three-tier Architecture

The overall three-tier ABC Architecture is illustrated as follows:

![Diagram of ABC Three-tier Architecture]

**Figure C.4[12.a], ABC Three-tier Architecture**

Beginning from the client, both C++ and Java are supported. Enhancements are made in C++ architecture but largely it is the same as in the two-tier architecture.

A new Distributed Data Service (DDS) is used which is a replacement for the std. Database networking software quite common in two-tier architecture. In the three-tier
architecture, the application server conducts the database management where users can pool the resources and the connections on a larger scale. Remote method invocation is also supported as a enhancement by the three-tier C++ client. A thin-client interface package is granted for Java clients, which allows the universal clients to work with component objects and to have access to the new features projected by the three-tier architecture. The server business objects in classic CORBA architecture are returned to the clients. Every called method results in the round-trip of a network. A lot of hand coding is required for a workaround to this. Due to which the purpose of the objects is defeated.

As this kind of architecture was undesired for the development purposes by the developers, ABC architecture allowed the client-based Business Component Objects to define and to get map to the server business objects. This kind of mapping is done automatically and so the clients cannot tell the difference between the client-based business component and the server business objects. To synchronize, the client and the server utilize A workspace Interchange service.

Philosophically, the adoption of an Internet protocol standard and IIOP would continue while the DCOM might not be able to achieve. But on the other hand, the tremendous growth of Windows2000 and its cluster software has make it a important platform for small and medium sized server requirements. The DCOM support would make sense only if we strict to Microsoft Domain. A structure is provided by this architecture for plugging into various middleware components.

There are the three major components on the server side: Distributed Service coordinator (DSC), application servers, and database servers. Within the application servers, one or more standard object services can be run. Majority of the servers will run at least the Persistent Object Service, the credential management component of the Security Service, workspace Interchange Service (for Java clients), Event Service, and the transaction Service.

For only single application server, all of these services will run within the same server along with the application server business objects but relying only on a single
application server might not work for many applications when the users are large. For any situation that involves hundreds of users if not thousands, the server software must possess the means to scale as demands. Generally, it must have the ability to start additional application servers over one or more host computers to effectively handle the demand when the resource increases.

ABC Architecture consists of a central component to provide this functionality which is the Distributed Service Coordinator (DSC). For each host running the ABC object services it must have at least one instance to run at all times when needed. The DSC will automatically start a new server if due to a heavy workload one is not available.

On a higher level, ABC three-tier architecture consists of these services and components.

**Thin Client Interface**

The functionality of an application server is completed if it is able to manage the business objects on the server side. Which in turn leads to a three-tier model where graphical user interface resides on the client and the application services, the database services, and the business objects resides on the server-side.

ABCs three-tier architecture has now several advantages such as, the distributed applications can be written using the business model without concerning the back-end databases. For e.g. JDBC is promoted for accessing the data in Java applications and applet. This kind of model forces the program to deal directly with database, which is acceptable for some systems but some of the applications at the enterprise level must be recoverable, scaleable, and need to use the logic based on server which is quite a inadequate approach. Due to several reasons, the two-tier systems cannot address such kind of reliability and scaleability issues so to extend this architecture over a wide area network makes it worst.

Figure below illustrates the distributed workspace architecture, which supports a new object known as business component so for e.g. if a client application is Java
program then these business components would be the Java Bean Objects which stands for the business objects on the server.

This means that a business client component contains all or part of the data of its counterpart business server object and a advantage of the component is that it can support both remote and local methods. Use of local methods is to fetch the local data elements and provide application-side business logic. And invoking methods, which are remote, can be forwarded to the CORBA object proxy that results in requests to the server by the standard IIOP method. Object Schema Compiler automatically generates the method bindings between the Component and the CORBA object.

![Diagram](image)

**Figure C.5[12.c], Thin Client Architecture**

The workspace architecture is strongly integrated with a distributed transaction service which provides for the perfect marriage of object management, distributed systems, and object persistence that provides a fault tolerance and robust infrastructure.
Deployment Architecture

The Server Deployment Architecture consists of base classes and a set of runtime services that implement server configuration, monitoring, and management. In this service, the management component provides a robust activation and deactivation service. And is also used for assigning transaction-begin requests with the proper server.

The Appendix B has presented a technical overview of ABC architecture, which is an extensive distributed object management architecture that has faced and tackled the toughest problems facing distributed application system. From the very beginning of the design phase, the different methodology and the tools exist to leverage this computing architecture into making the most of what object technology has to offer in today’s world of Internet Computing Architecture. The ABCs architecture has helped in making these complex services easy to use and demonstrated the real power of distributed object world.
REFERENCES


3. Client/Server Architecture by Alex Berson, McGraw-Hill Companies, April 1996


5. Java Distributed Computing by Jim Farley, Chapter 3, Distributed Objects, O'Reilly and Associates, 1998


8. (a)Figure 1: Network Computing Architecture, (b)Figure 2: Java and Network Computing Architecture, (c)Figure 6: Java product matrix; Bringing Java to the Enterprise, An Oracle Technical White Paper, June 1997

10. Figure (24(a), 25(b), 26(c), & 62(d)), Enabling the Information Age Through Network Computing by Daniel Sagalowicg; http://www.oracle.com


12. (a)Figure 34.4: The generic application server framework architecture; (b)Figure 34.3: The Object Transaction Monitor Architecture; (c)Figure 14.4: The ORB architecture; Building Java Enterprise systems with J2EE by Perrone and Chaganti; Sams ;2000

13. Figure 1: Windows DNA tools and system services; Introducing Windows DNA: Framework for a New Generation of Computing Solutions; Microsoft Corporation; January 1999; http://www.microsoft.com

14. Figure 3: Two-Tier Implementation; Designing Efficient Applications for Microsoft SQL server by Peter Hussey; August 1997; Microsoft Corporation

15. Oracle 8i by Michael Abbey, Michael J. Corey, and Ian Abramson, Chapter 1,3 and 5, McGraw-Hill, 1999

16. IBM Web Sphere Starter Kit by Ron Ben-Natan and Ori Sasson, Chapter 8 and 9, McGraw-Hill, 2000