Component Technology : An Overview

Saptarshi Mondal*

The Internet started out as a project to connect scientists and researchers around the world and grew into a general-purpose communication tools. The phenomenal growth of Internet and expansion of corporate Intranet throughout the world has resulted in an epidemic of browser mania. Conventional business application are giving way to browser based application. With the advent of internet and E-commerce the software industry has also witnessed an acceleration of new high tech development with latest tools and programming paradigms which leads to the emergence of a virtually universal platform that is not only suitable in the current scenario but also seamlessly adaptable to future hardware and software development, the web browser. HTML interpreters have been written for all operating system imaginable, even telephones. The reality of an Internet ready toaster doesn't seem so comical anymore. HTML is here to stay for a good while and make perfect sense as a "Platform".

Today's customers are webs enabled and spoiled to near instant communication and transaction. If a web site doesn't load in a few seconds a few impatient mouse clicks land the user in one that does. The incentive to adopt a web browser as a universal GUI is obliquity; virtually everyone with a PC has at least some incline of browser based document display and navigation. Thus the information technology managers expect to reduce the user training cost and give access to everyone of the organization direct access to different business applications. Another goal of browser based application is to move their execution to a combination of web and application server eliminating the need to distribute new and updated application to the desktop and laptop clients. As a matter of fact this new digital nervous system has already hinted enough of its grips over the future.

Windows Distributed Digital Application Architecture (DNA) is Microsoft's implementation layer of digital nervous system. The heart of Windows DNA is the integration of web and client/ server application development model through a common object model. Windows DNA is a synonym of "COM everywhere". The common object model (COM) or Distributed COM (DCOM) is the "unified way to interconnect the component of windows DNA". Microsoft promotes DNA as a robust framework for building scalable, multitiered Internet application. To a developer this framework is more like a road map of Microsoft's tools and products and where they fit in multitiered design.

Browsers with all of their cosmetic multimedia glories are for all practical purposes dumb read only viewers. They are stateless and non-linear i.e. they can jump from one page to another randomly without following any predefined path. Keeping state has never been a problem with traditional Windows applications that are event driven and follow a finite state machine pattern. But HTML & DNA force developer to think about statelessness. To complicate the matter DNA application user (web browser) have fewer constraints placed on their working environment and are free to roam about unchecked. Theoretically browser is like a non-deterministic finite automata with every state a starting and final state. Thus the browser is forced with the problem of tracing a seaminglessly anonymous connection as it randomly hops about from one page to another. One of the possible solution of this problem is mapping the IP address and using timers for creating session and maintain of state for each browser accessing the server.

* Final Year Student of Computer Science & Technology

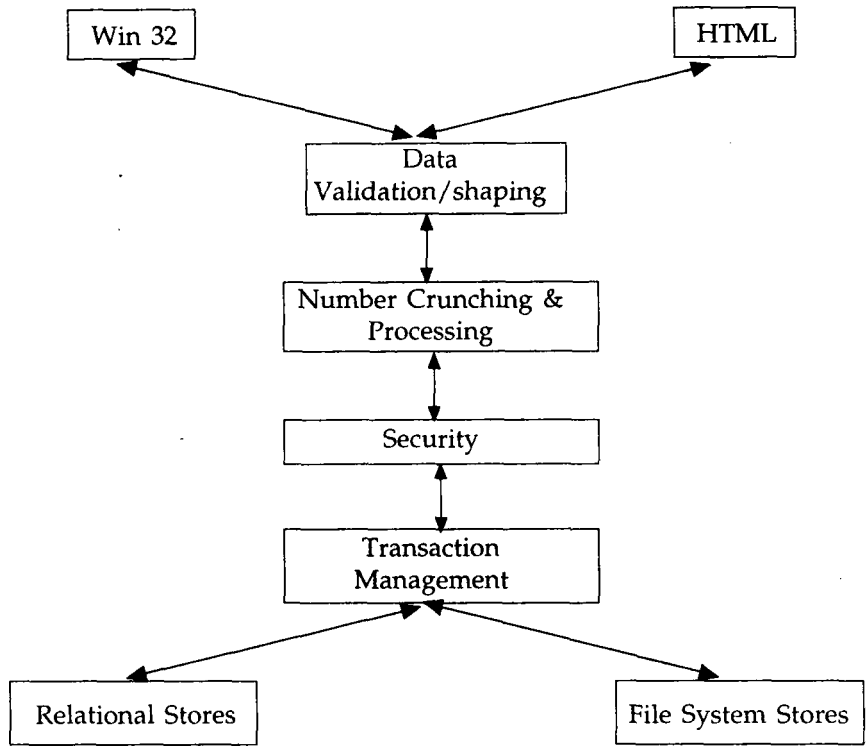


Fig-1 : Multitiered web based project

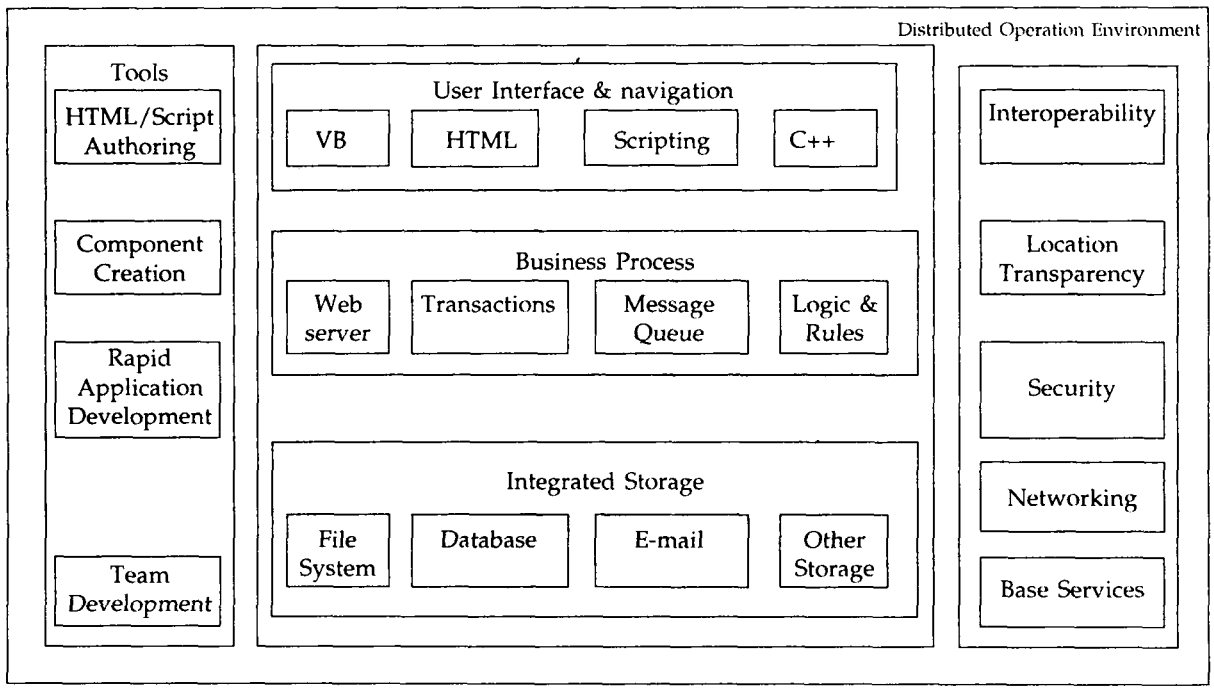


Fig-2 : DNA framework

From a high level perspective the association of technologies within DNA framework can be described by the following fig–2. In order to make DNA work, there must be support from the operating system Kernel. An authority service must be present at all times to interpret and oversee communication protocols and component synergy. This is where COM/DCOM comes into the DNA picture as the distributed operating environment. The following two items compose the Distributed Operating environment.

a. **Interoperability :** For component technology to work a consistent process of binary capability discovery must exist. Component should be able to query other components and through a process of elimination find whether the requested behavior exists.

b. **Location transparency :** Component in a distributed environment like DNA must not care about the actual physical location. In COM the actual location of the component are written in windows registry which enable DNA application to execute as if they are in their local environment even though the area spread out all over the internet.

There are many services such as security, networking, thread management, transactions.

Synchronization support, component registration, debugging and so on which are required to make component work these services are essential for providing a backdrop in which component can be free to move about without concern of networking or operating system implementation. A COM compliant component can incorporate in heterogeneous environment simply by making calls to the base service API and lets the operating system do the rest.

One of the most successful models emerges from the client/server world has been the three-tiered model. A tier is a collection or set of independent homogeneous object that together solve a large but common problem. The three tiers are generally known as :

a. **Presentation :** This involves all interactions with the user. The model doesn't allow interaction with the user at any other tier. Doing so couple the lower layers with this responsibility and render the model inflexible with time. The presentation tier has a defined set of interface that enable it to communicate to the business layer, and that is the only capability it has.

b. **Business Logic :** Here most of the processing is carried out. All business specific rules are grouped into this tier. This is part of the application that actually solves the problem. It is in the middleware between user and any physical database. It also should not know specific details of data service tier below, nor the type of presentation above. It should only process data, not store it or present it.

c. **Data Services :** Specific data service mechanism like low-level database access or SQL comprises this layer. An application doesn't need require a database from a three-tiered model. Any kind of persistent storage can be placed here (file system, e-mail, multimedia stream, etc) away from the presentation and logic layers so as to increase the potential for maintainability and evolution of the system in future.

As application evolves and becomes more complex, it is some time necessary to break a particular tier into several pieces. This result in a multitiered, or n-tired architectures. The following fig–1 schematically represents a complex in-tier web-based project.

As described by the Microsoft the core component technology comprises of the following :

1. The **Component Object Model (COM)** : The underlying distributed object model for all COM components. This includes :

- Distributed capabilities, commonly referred to as **DCOM.**
- The **Service Control Manager :** A part of the COM library responsible for locating class implementations implemented as libraries, local processes, or remote servers.

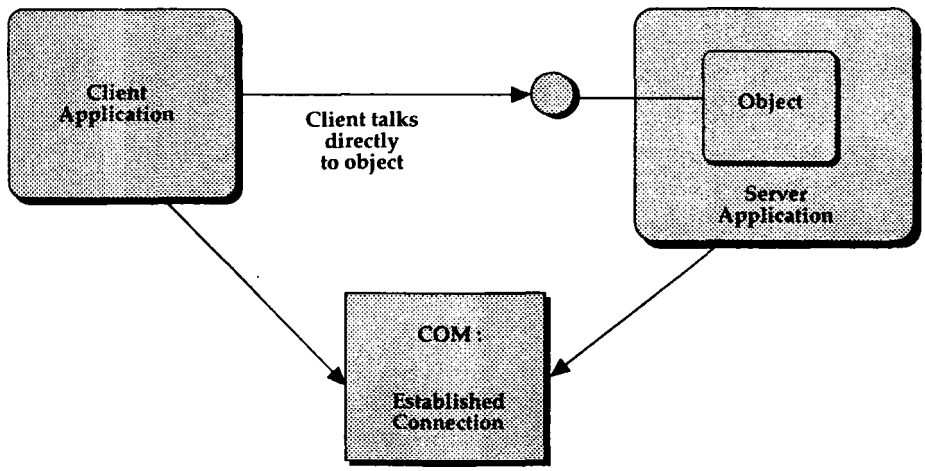
- **Security :** A rich and pluggable infrastructure for building secure distributed applications. Facilities are provided for authentication, authorization and privacy.
- **Structured Storage :** Provides a rich, transaction based, hierarchical file format that enables COM applications to create files that can be shared across applications and platforms.
- **Monikers :** Allows references to objects to be stored persistently. Provides for persistent, intelligent names.
- **Automation :** Allows objects to expose functionality to high-level programming languages and scripting environments.

2. **MS-RPC :** An implementation of the Distributed Computing Environment (DCE), Remote Procedure Call (RPC) specification, upon which COM is based.

3. The **Registry :** Provides a database of COM components and their configuration information.

4. The **Security Support Provider Interface (SSPI) :** A standard for pluggable security providers.

5. The **Windows NT Distributed Security Provider :** An SSPI security provider, which supports the Windows NT Distributed Security model (also called the NTLM SSP).



As with hardware developers and the integrated circuit, applications developers now do not have to worry about *how* to build that function; they can simply purchase that function. The situation is much the same as when you buy an integrated circuit today: You don't buy the sources to the IC and rebuild the IC yourself. COM allows you to simply buy the software component, just as you would buy an integrated circuit. The component is compatible with anything you "plug" it into.

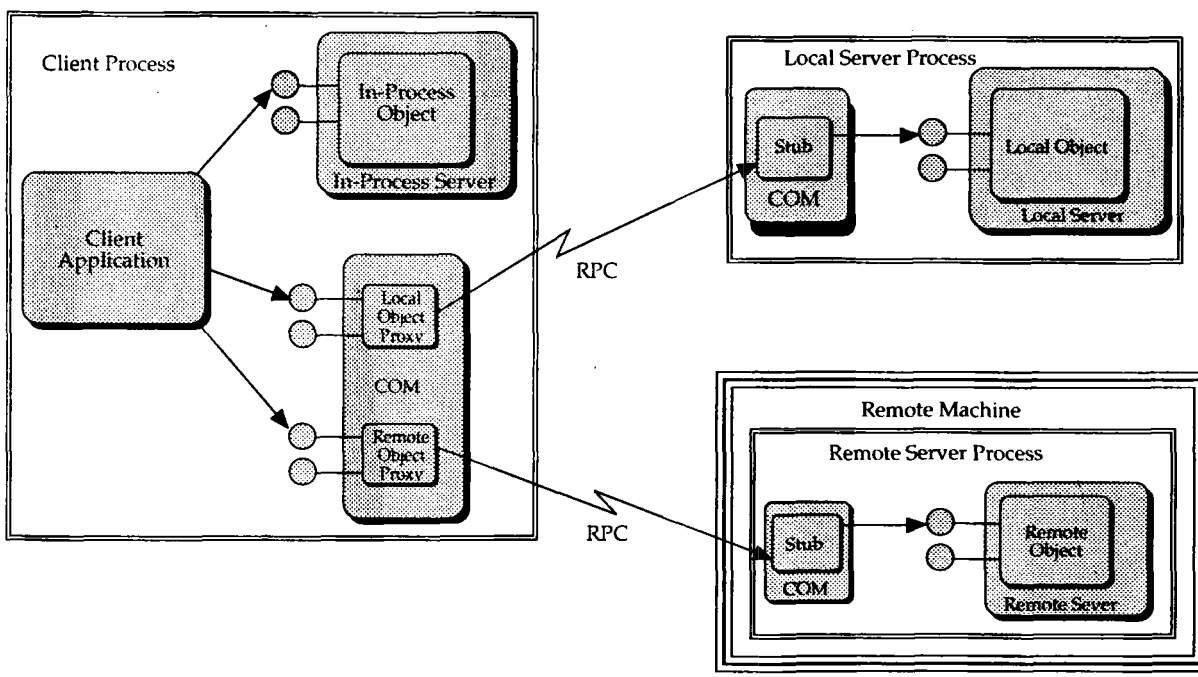
By enabling the development of component software, COM provides a much more productive way to design, build, sell, use and re-use software. Component software has significant implications for software vendors, users and corporations :

- **Application developers :** are enabled to build and distribute applications more easily than ever before. Component objects provide both scalability from single process to enterprise networks and modularity for code reuse. In addition, developers can attain higher productivity because they can learn one object system for many platforms.
- **Vendors :** are provided with a single model for interacting with other applications and the distributed computing environment. While component software can readily be added to

existing applications without fundamental rewriting, it also provides the opportunity to modularize applications and to incrementally replace system capabilities where appropriate. The advent of component software will help create more diverse market segments and niches for small, medium and large vendors.

- **End-users:** will see a much greater range of software choices, coupled with better productivity. Users will have access to hundreds of objects across client and server platforms—objects that were previously developed by independent software vendors (ISVs) and corporations. In addition, as users see the possibilities of component software, demand is likely to increase for specialized component they can purchase at a local software retail outlet and plug into applications.
- **Corporations:** benefit from lower costs for corporate computing, helping IS departments work more efficiently, and enabling corporate computer users to be more productive. IS developers will spend less time developing general purpose software components and more time developing "glue" components to create business-specific solutions. Existing applications do not need to be rewritten to take advantage of a component architecture. Instead, corporate developers can create object-based "wrappers" that encapsulate the legacy application and make its operations and data available as an object to other software components in the network.

In sum, only with a binary standard can an object model provide the type of structure necessary for full interoperability, evolution, and re-use between any application or component supplied by any vendor on a single machine architecture. Only with an architecture-independent network wire protocol standard can an object model provide full interoperability, evolution, and re-use between any application or component supplied by any vendor in a network of heterogeneous computers. With its binary and networking standards, COM opens the doors for a revolution in software innovation without a revolution in networking, hardware, or programming and programming tools.



Component technology is an emerging field through which future of software industry is going to witness a totally new paradigm of tools, products and of course software. In this emerging technology it is not only the Microsoft but also other vendors are introducing their own implementation of component technology such as CORBA (Common Object Request Broker Architecture) which established its firm grip in the Unix world and directly compete with COM/ DCOM. The ever-increasing need of scalable, robust and complex software is the driving force behind this new technology and this would surely squish out the offered potential out of this.