Possible Futures of Application Architectures

Application architecture is the key enabler—and key limiting factor—of the value an IT product or service can provide. Recent developments in the field of application architecture promise tremendous improvements in productivity and quality. These developments are embodied in the two competing Internet-focused application frameworks, Sun's Java 2 Enterprise Edition and Microsoft's .Net platform. Both of these platforms promote the adoption of the Web Services paradigm, a "disruptive technology" that is changing the face of software development. Electronic commerce and communities will be transformed over the next few years in ways we have yet to comprehend.

Software architects, entrepreneurs, and investors seeking to exploit the possibilities of the ongoing growth in the IT arena need to understand what application architecture trends are currently operative and how they will affect the shape of the IT marketplace over the next several years. Using Scenario Planning, a time-honored prognostication technique, this paper will provide the reader with a solid understanding of the new developments in application architecture.

A South Wind Research Report

South Wind Design, Inc.
Ann Arbor, Michigan

Last updated 1/4/2002 5:26 PM

Table Of Contents

Figures

Tables

Possible Futures Of Application Architectures

Application architectures have been growing in diversity, richness, and complexity for well over forty years now, since the development of the first operating systems (OS) and high-level languages. However, over the last two decades the marketplace has seen a true explosion in architectural possibilities, with many factors contributing, including but not limited to the following:

- The emergence of the personal computer
- The development of windowed user interfaces
- Local- and wide-area networks (LANs and WANs)
- Convergence of the telecommunications and data connectivity infrastructures, extending even into personal residences in most metropolitan areas worldwide
- The popularization of the Internet as a text-based communications medium (electronic mail, file sharing protocols, and newsgroups)
- The emergence of electronic communities like CompuServe, Delphi, and America Online
- The dramatic growth of the World-Wide Web
- The establishment of wireless networks using cellular and other similar technologies
- The ongoing refinement of best practices in software design and development
- The rise and continuing dominance of industry giants such as Microsoft, Oracle, and Sun, and the continuing evolution of earlier-era giants like IBM and AT&T
- The emergence of the Open Source movement
- A tremendous increase in processing power
- An equally remarkable increase in capacity and access speed, coupled with a proportionate decrease in cost, of both magnetic and optical storage
- The emergence of Internet-focused application frameworks—Java 2 Enterprise Edition (J2EE) and Microsoft.NET—that by their very nature dictate many aspects of application architecture

Application architecture is the key enabler—and key limiting factor—of the value an IT product or service can provide. It determines the structure and function of the media to be used in connecting sources of digital value to the consumers of that value. Software architects, entrepreneurs, and investors seeking to exploit the possibilities of the ongoing growth in the information technology arena need a working knowledge of application architectures.

THE FUTURE STATE OF APPLICATION ARCHITECTURES IS BY NO MEANS CAST IN STONE.

The future state of application architectures is by no means cast in stone. If the Internet revolution continues, either J2EE or .NET may become the framework of choice, with radically different implications for how applications are structured. Conversely, if economic conditions worsen and organizations can't justify new infrastructure investments, the Internet revolution may stall, and something akin to the current status quo may prevail for some time to come.

Biological View of Architectural Evolution

Computer applications have undergone—are undergoing—an architectural evolution that has direct parallels with the physiological evolution of biologic entities. External events drove the activities of the earliest living things, whose analogs in today's world are viruses. Gradually, the development of richer internal processing led to the centralized decision-making capabilities and proactive operations we see in multi-cellular life forms.

Paradoxically, though, as life forms evolve further a decentralization and distribution of decision-making power and storage of knowledge leads to a more robust and resilient species. All these same evolutionary forces are now visibly at work in the realm of application architecture.

Successfully navigating this revolution requires that we understand how evolutionary processes work. Nature pursues multiple branching possibilities simultaneously, and only time will tell which branches die out and which one dominates.

We need to be clear on what it is that is evolving here. Application architectures are not ends in themselves, but are the means by which an organization exploits the computing technology at its disposal. The network connectivity and user interfaces of the organization's IT assets are its sense organs and nerve endings, and the architecture of the organization's enterprise applications is the anatomy and physiology of its nervous system. Just as more sophisticated, responsive nervous systems characterize higher-level organisms, the increasing complexity and sensitivity of enterprise applications characterize advanced organizations.

Scenario Planning

To address the problem of uncertainty in strategic planning, large enterprises developed a powerful prognostication tool called Scenario Planning in the last half of the 20th Century. Royal Dutch Shell, one of the early adopters of Scenario Planning, used it to leverage their market position from eighth to first place between 1970 and 1985. As an illustration of the diverse uses to which Scenario Planning has been put, the technique was also employed by the transitional government of South Africa to make their successful transition away from an apartheid-based society in the early 1990's.

While not pretending to such lofty goals as these, this paper uses the Scenario Planning technique to answer the following question:

What application architecture trends are currently operative and how will they affect the shape of the IT marketplace over the next several years?

We include some background material on Scenario Planning toward the back of the document, in Appendix: Introduction To Scenario Planning. In addition, several good pointers are given to additional reading in the References section. But in a thirty-second summary, the way scenario planning works is by following the steps shown in the following table:

Step	*Description*
Frame the question	Develop a precise definition of what it is you want to know. Include a time frame, geographic scope, or any other relevant limitation to keep a sharp focus.
Research	Learn everything you can about the context of the question. Make sure you absorb contrasting and competing views, not just your favorite viewpoint.

Step	*Description*
Identify local forces	Sift out the factors that affect the shape of the possible answers to the question.
Identify predetermined elements	Some of the factors have a high degree of certainty: demographic projections, tidal forces, etc. These will evolve in the same way regardless of the turn other variables in the situation may take.
Identify critical uncertainties	Many factors will affect the outcome of the question. Identify the most influential forces, preferably two or three at most.
Develop a matrix of critical uncertainties	Plot the critical uncertainties against each other in a combinatorial matrix.
Identify and name the scenarios	Eliminate absurd combinations and combine those that will lead to nearly identical outcomes, reducing the combinations to a set of distinct outcomes—scenarios describing possible futures. Give each of the scenarios a catchy name.
Develop plot lines	Real-world scenarios tend to follow common story lines: winners and losers, challenge and response, evolution, revolution, the Lone Ranger, etc. Many technology scenarios are evolutionary, but every so often a revolution happens, a paradigm shift.
Flesh out the scenarios	Although the characters in planning scenarios are institutions and global forces rather than individuals, the implications of the scenario are often easiest to grasp when cast in a personal light. Create a plausible, interesting, and informative story around each scenario.
Present the scenarios	The purpose of planning scenarios is to influence decision-making, so the scenarios will need to be explained and defended to the decision-makers—strategic management, investors, key customers, etc. Try to describe each scenario dramatically enough to give it life in the minds of the audience.

Since we have already stated our question above, the next few sections describe the features and operating forces of the world of application architectures. Then we'll move on to the scenario-building process.

Dimensions Of Architectural Variation

The above-listed array of factors has many possible permutations and facets. To make sense of application architecture trends, we need to organize the factors along meaningful dimensions. In real life these dimensions are continuously and simultaneously operational, of course, so we must always keep in mind that our limited focus at any given time is providing only a part of the picture.

We'll examine each of the following dimensions in turn:

- Connectivity
- Networking Foundation
- Application Framework
- Data Storage
- Agency Versus User Control
- Structural Model

Connectivity

This dimension refers to the ability (or lack thereof) to interconnect the various architectural components, such as the clients and server in a client/server architecture. Two facets exist to the connectivity dimension: bandwidth and continuity of connectivity. Standalone applications do not participate in this dimension; the matrix below illustrates the connected possibilities.

Table 1 - Continuity Of Connectivity vs. Bandwidth Matrix

	Bandwidth		
Continuity of Connectivity	**Broadband (1MB+)**	**Wideband (56K-1MB)**	**Narrowband (less than 56K)**
Continuous	- Corporate desktop PCs & servers connecting via T3, T1, ISDN, DSL	- Corporate desktop PCs connected via dial-up - Bluetooth-enabled devices - Web-enabled cellular phones (2.5G service)	- Web-enabled cellular phones (2G service)
Intermittent	- Mobile laptops or home-based PCs connecting remotely via cable modems, ISDN, or DSL - 3G cellular service	- Mobile laptops or home-based PCs connecting remotely via land-line dial-up and ISDN - Wireless LANs	- Palm VII with integrated cellular connection - PDAs with modems

The broad spectrum of possibilities should not be construed with any sense of judgment. High bandwidth and continuous connectivity are a requirement for some types of applications, such as multimedia-based online learning courses. However, many other

market niches are consistent with an application architecture that is intermittent, low bandwidth, or both. "Worse is better" in these cases, which include such broad activity areas as messaging and remote data entry. These media fit Marshall McLuhan's definition of "cool media" because of their low definition and highly participative nature.

MESSAGING AND REMOTE DATA ENTRY FIT MARSHALL MCLUHAN'S DEFINITION OF "COOL MEDIA" BECAUSE OF THEIR LOW DEFINITION AND HIGHLY PARTICIPATIVE NATURE.

We assume that any non-trivial new application developed in the next five years will be pulled in the direction of providing support for most, if not all, cells in this matrix.

Networking Foundation

TCP/IP

Application architectures that involve connectivity have largely standardized on TCP/IP based connectivity in the last decade, supplanting proprietary and/or non-routable protocols such as NetBIOS, IPX/SPX, and Banyan Vines. The latter group of protocols still have significant market share, though, especially IPX/SPX, which is the foundation of earlier-generation Novell Netware networks. IBM's SNA protocols are also still in heavy use in environments that depend on mainframe servers.

In the TCP/IP arena, there are two main flavors based on the two main server OS families in use today: Windows and Unix. The Unix camp includes proprietary versions as well as freeware, stemming from three main sources: the original Bell Labs version; the UC Berkeley's BSD version; and the Open Source movement's GNU/Linux OS kernel. The three flavors of Unix are equivalent in most technical details, differing almost exclusively in their licensing models. Microsoft Corporation holds proprietary control over the Windows family, which now includes three generations of Windows NT and the recently released Windows 2000 versions. Linux is by far the fastest growing option in the Unix camp, approaching Windows in popularity with respect to new OS installations.

HTTP

Another standard that has emerged, for better or worse, is the Hyper Text Transfer Protocol, or HTTP, which is the application-level communications protocol of the World Wide Web. HTTP is a connectionless client/server protocol that by convention is allowed access through corporate firewalls.

HTTP was originally designed for the dissemination of relatively small text documents containing hypertext markup (most often using the Hyper Text Markup Language, or HTML). However, its firewall-friendliness has led to its use as a ubiquitous means of circumventing firewall restrictions, a practice referred to as "HTTP tunneling."

The HTTP specification was reasonably well-written to begin with and any ambiguities have been worked out in the millions or billions of usage hours it has undergone. It has a number of limitations, however:

- It uses a relatively small packet size that utilizes the underlying network efficiently for small text documents, but not so efficiently for the huge binary objects now being transported via HTTP.
- Its connectionless nature makes it less suitable for applications that are inherently connection-oriented, such as groupware and document-oriented office productivity applications
- Its inherent client/server architecture makes peer-to-peer interactions clumsy and unreliable

Application Framework

Applications are almost never developed from scratch. Instead, they rely on existing application frameworks to provide an architectural starting point and a library of reusable components. In addition, many application frameworks offer development tools that boost productivity in multiple ways. Some higher-end frameworks go far beyond this to offer business process modeling and re-engineering features that are designed to take an enterprise to a new level of organizational integration in a short period of time.

Application frameworks boost productivity and quality in most cases, but there is a price to pay. The framework you select will limit many of the architectural options by imposing solutions in various aspects of the design. Sometimes these impositions can be circumvented, but when this is possible it usually involves significant additional effort and risk. If the imposed architectural features are neutral or beneficial to the application, as they usually are when an appropriate framework is selected, the application framework's benefits outweigh its costs.

Three types of application frameworks are prevalent at this point:

- Integrated development environments (IDEs; e.g., Visual Studio, Delphi, etc.)
- Enterprise application frameworks (e.g., SAP, PeopleSoft, etc.)
- Internet-focused frameworks (e.g., J2EE and Microsoft.NET)

As listed, these are in *decreasing* order of prevalence and *increasing* order of future importance. In other words, IDEs are the most common framework and is expected to have the least influence on future development. Contrariwise, Internet-focused frameworks are just starting to be deployed but are expected to dominate the application architectures of the next decade.

Integrated Development Environments (IDEs)

The least expensive application frameworks are general-purpose 3GL and 4GL language toolkits called integrated development environments (IDEs), such as the following:

- Microsoft Visual Studio 6 (Visual C++, Visual J++, Visual Basic, Visual Interdev, etc.)
- Inprise/Borland's Delphi, C++ Builder, and Jbuilder
- Symantec Visual Café

- Sybase PowerBuilder

INTEGRATION OF A DISTRIBUTED APPLICATION BUILT USING AN IDE OCCURS ON PAPER FOR THE MOST PART, IN THE SYSTEM DOCUMENTATION.

Each of these offers many tools that simplify development of standalone applications. Current-generation IDEs are often used in a client/server environment to produce client-side and server-side components as separate applications; the integration of a distributed application occurs on paper for the most part, in the system documentation.

IDEs target particular environments; most currently target either Windows or Java environments. Each MS Windows-targeted IDE has the capability to employ common components such as ActiveX controls. Most IDEs also allow you to build such components as well. Java IDEs such as JBuilder and Visual Café allow you to use and build Java Beans, which are Java-specific equivalents of ActiveX components.

The chief disadvantage of IDEs is that they are not generally very useful for producing distributed architecture applications, which are the most common new development projects being undertaken as of 2001. Integration of distributed application components, which may include Web pages, server-side executable components, and client-side components along with configuration and data files, is a manual process with such IDEs. MS Visual Interdev does provide integration for all of its components, but it is overly Microsoft-centric in the client and server environments it supports.

Enterprise Application Frameworks

This family has several major players that each come from different starting points, but have converged on a single goal: providing a comprehensive IT framework for medium- to large-scale enterprises. The major players include:

Table 2 - Enterprise Application Framework Providers

SAP R/3	SAP is a German firm that started out in the financial and materials planning arena.
PeopleSoft	Originally a human resources system.
J.D. Edwards	Originally an accounting systems developer.
Oracle	Oracle has a diverse suite of enterprise applications based on its flagship relational database (RDBMS).
Baan	Baan originated as a financial and administrative consulting service provider. It was catapulted to international prominence in this market when it obtained a ~$150M USD contract with Boeing in the mid-1990's.
Broadvision	Broadvision is unique in having originated in the e-commerce marketplace. It is used in many large-scale commercial sites in the B2C arena.
Siebel	Siebel was started by an ex-Oracle sales giant, initially focusing on

Systems	sales force automation, now dominant in the field of Customer Relationship Management (CRM).

ENTERPRISE APPLICATION FRAMEWORKS PROVIDE A COMPREHENSIVE IT FRAMEWORK FOR MEDIUM- TO LARGE-SCALE ENTERPRISES.

Each of these enterprise-scale application frameworks has its own development tools and techniques; each imposes many constraints on the shape of the application architecture, at the same time providing many benefits through the encapsulation and complete integration of best-practices business processes. Each framework offers some flexibility in the architecture of end-user interfaces; in particular, all offer some sort of Web integration for use in intranet, extranet, and Internet applications.

Internet-Focused Frameworks

General-purpose frameworks are emerging that are targeted at Internet development activities. There are two proprietary frameworks and one Open Source framework. Each is backed by an industry heavyweight and both can be expected to garner significant mind and market share. They are shown in Table 3 - Internet-Focused Frameworks.

Table 3 - Internet-Focused Frameworks

Platform	Intellectual Property Status	Implementation(s)
Java 2 Enterprise Edition (J2EE)	Open Source (in most cases with proprietary extensions)	BEA WebLogic, IBM WebSphere, Lutris Enhydra, numerous others
Oracle 9iAS (J2EE-based)	Combination of Open Source and proprietary features	J2EE-based, but largely proprietary implementation package consisting of Oracle8i database servers, Oracle9i Application Server, Oracle XML extensions, JDeveloper and other IDE tools, and legacy Oracle tools such as Oracle Forms and Reports; existing systems, such as enterprise application frameworks, can also be easily integrated
Microsoft.Net	Proprietary	Microsoft Visual Studio.Net aod Microsoft.Net SDK (Both now in beta)

INTERNET-FOCUSED FRAMEWORKS INTEGRATE MANY USEFUL SERVICES FOR DEVELOPMENT OF DISTRIBUTED APPLICATIONS.

The J2EE framework is supported by the R&D efforts of industry leaders IBM and Sun in addition to the tens of thousands of Open Source volunteers. This family is based on the Java language, especially the Java 2 Enterprise Edition (J2EE) framework. Although Open Source implies free software, J2EE IDEs range from freeware to expensive, high-value-added commercial tools from

heavyweights such as BEA and IBM. Oracle's offering is also J2EE-compliant (albeit without abandoning its user base invested in earlier distributed frameworks), so it is reasonable to say there are only two major Internet-focused frameworks.

Internet-focused frameworks integrate many useful services for development of distributed applications, including but not limited to the following:

- HTTP server
- Partitioning of application functionality across multiple server machines, including any combination servers
- Database integration
- XML parsing, transformation, and DBMS integration
- Naming and directory services
- Message queues, including electronic mail, Simple Message Service, and other asynchronous communication protocols
- Remote procedure calls support including service discovery, description, and invocation
- Web Services (a specific type of remote procedure call expected to become the standard model for e-business interactions; see the Web Services section under Structural Model/Application Server Architectures for more details)
- Wireless and other non-traditional device access via separation of content and presentation
- Workflow

As we mentioned earlier, .NET and J2EE represent the most important new features of the application architecture landscape as of 2001. Each is expected to grow dramatically in the next few years, but it is unclear which platform will dominate the scene.

Data Storage

Applications store transient data in memory or in temporary files that are deleted when the application session closes. To maintain application state between sessions, data must be stored somewhere, almost always on magnetic disk. The disk storage unit may be physically attached to the computer (workstation or server) on which the application is running, or it may be a shared location on a network server.

If multiple applications store data in a central location, serialization issues inevitably arise. Simply put, serialization is the requirement that data storage and retrieval operations make consistent sense over time; for example, if multiple ticket agents are assigning seats on an airline, no two passengers should be assigned the same seat for any given flight. There are many ways for serialization to go awry, most of which are too complex to be explained in a paper of this brevity, and all of which have happened at some point in the sordid history of computing. As a means of resolving these issues, three different types of solutions have arisen:

1. File systems, in which the unit of consistency is the file or some physical division thereof.
2. Database management systems (DBMS), in which the unit of consistency is a logically defined object whose physical state is maintained by the DBMS. Two kinds of DBMS are in more or less common use at this point: relational database management systems (RDBMS) and object-oriented database management systems (OODBMS). Without getting into the details of the differences between the two, suffice it to note that RDBMS have held sway in the marketplace for the last 15-20 years, and are therefore more mature than OODBMS as a general rule. On the other hand, OODBMS are arguably better suited architecturally to storage of the increasingly complex data types defined by formats such as XML (eXtensible Markup Language). Consequently, many expert observers expect OODBMS to move from CAD applications, their current market niche, into wider use in the near future.
3. Content management systems (CMS), are a marriage of OODBMS with version control systems (VCS), in which the unit of consistency is the state of a file at a given point in time; i.e., multiple versions of the same file may be maintained that are essentially different generations of the same logical entity. CMS also typically incorporate workflow control mechanisms.

The data storage issue appears to be moving toward a resolution, although it may be a decade or more until it fully consolidates. RDBMS systems are investing their tremendous cash hoards in R&D efforts that extend DBMS functionality into each of the above areas.

Oracle, for example, has release its Internet File System (IFS), which provides many advanced file system features in addition to Internet accessibility. Oracle is also expected to increase its content management capabilities dramatically over the next five years, to the point where it will offer serious competition to CMS market leaders Documentum and XYEnterprise. The Open Source movement's MySQL RDBMS can likewise be coupled with the version control system CVS, another Open Source effort, to form something resembling a CMS, albeit without the smooth integration to be expected of Oracle's products. Microsoft has an initiative, code-named Tahoe, that will soon result in a document-management offering as part of its .NET initiative.

ORACLE'S INTERNET FILE SYSTEM (IFS) PROVIDES MANY ADVANCED FILE SYSTEM FEATURES IN ADDITION TO INTERNET ACCESSIBILITY.

A trend worth noting in the data storage arena is the recent trend toward adding more and more intelligence to DBMS applications, particularly in the integration of domain-specific lexicons and ontologies as a "knowledge layer" operating above the DBMS. Tools such as Synaptica and Apelon offer the possibility of adding concept-based search to data and document bases, hinting at potentially great improvements in productivity and quality for the output of "knowledge workers" such as marketers, journalists, and analysts.

Agency Versus User Control

Although it has little direct effect on application architecture in practical terms, it is worth noting that distributed computing models foster the development of *agents*, software programs that operate without direct user control to accomplish predefined goals. Agents require a modicum of intelligence to cope gracefully with emergent situations without user intervention. DARPA has initiated a program to promote the development of agent technology, the first fruit of which is DAML, the DARPA Agent Markup Language.

Agent-oriented applications are expected to represent a significant new market niche in a wide variety of environments, especially as pervasive network connectivity is established. Agent applications have a significant advantage over user-controlled applications in a low-bandwidth environment in particular, since they have no need to carry the overhead of any sort of UI.

Structural Model

The final dimension we will consider is the structural model, the division of labor between components in the application architecture. Interestingly, in the current architectural explosion we see not only the continuation of existing and emergence of new architectural forms, but the re-emergence of architectures that had gone out of style decades ago! The accompanying figure shows the evolutionary context of the architectural structures described in this section, including the mainframe-based ancestors that are not otherwise mentioned in this paper.

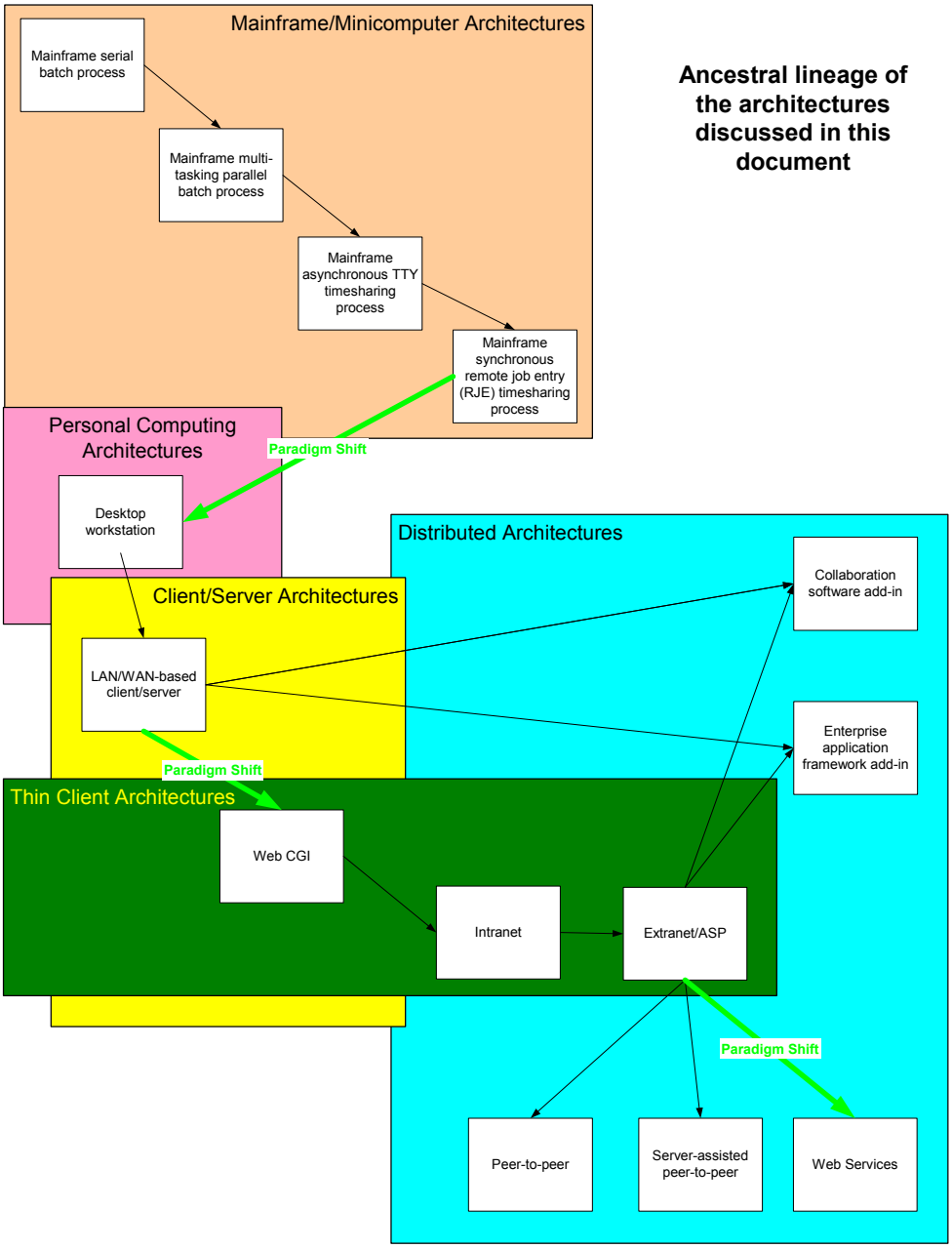

Figure 1 - Ancestral lineage of the architectures discussed in this document

Paradigm Shifts

There have been two paradigm shifts in the realm of application structural models: the shift from mainframe to desktop in the mid-1980's, and the shift from fat to thin client architectures in the mid-1990's.

The mainframe world had evolved from batch applications to a thin-client world where users updated mainframe data either by asynchronous, line-oriented text interactions entered on dumb terminals such as teletype or CRTs (a CRT was a cathode-ray tube with attached keyboard), or by synchronous transmission of mini-batch jobs created using

formatted data entry screens on smart terminals like the IBM 3270. The value from the sales perspective was in the mainframe hardware; other factors, like software, connectivity, and terminals, were often loss leaders used to cement the deal.

With the commercial emergence of the personal computer (PC) in the late 1970's, and especially the IBM PC in 1982, a new possibility emerged. End users could do meaningful work (document creation, accounting, graphic arts) on machines they themselves owned and maintained, which sat on or beside their desks. Whole new industries of application and peripheral development were created as the PC transformed the business workplace. The mainframe was still valuable, but as PC networking evolved the mainframe moved into a new servant role in which it provided secure storage, printing, and other centralized services, while the "real work" was done on the PCs.

With the privatization of the Internet in the early 1990's and the widespread availability of cheap modem-based connectivity, the scene was set for the emergence of the World-Wide Web, which provided universal access not just to prosaic mainframe-type services, but to the fast and cheap interchange of text and graphics, using protocols that allowed almost anyone to set up their own Web site. The addition of protocols for interactive applications over the Internet led to the second major paradigm shift, in which applications moved back from the desktop PC to the server, albeit a server that was now located on the Internet rather than a corporate LAN or WAN.

Because each layer has built upon rather than totally supplanting its predecessors, a rich and fertile environment now exists for application architecture. In today's world, many structural models exist and the architect is often free to choose the most suitable for any given application.

The following subsections describe what are currently the most prominent structural models and how they relate to each other.

Standalone Workstation

This is the architecture that epitomized the emergence of the personal computer (PC). It operates in standalone mode (i.e., not networked to other computers), typically on a stationary workstation. For obvious reasons, any resources used must be on the workstation itself.

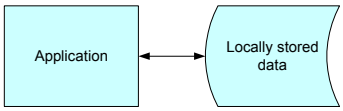

Figure 2 - Standalone Workstation Architecture

Client/Server

The client/server model is the typical application found in mainframe- and LAN/WAN-based environments. A "fat client" application running on one or more workstations relies on a server-based data store, typically an RDBMS accessed via the ODBC database connectivity standard. Although work appears to be distributed, business logic is often located almost entirely on the client, with the server used simply to provide serialization

CLIENT/SERVER BUSINESS LOGIC IS OFTEN LOCATED ALMOST ENTIRELY ON THE CLIENT, WITH THE SERVER USED SIMPLY TO PROVIDE SERIALIZATION OF TRANSACTION DATA.

of transaction data. When the server contains application code at all, it will almost always be related to data serialization.

Communication between client and server is normally accomplished using proprietary syntax and application-embedded semantics. Outsiders cannot "join in" on the conversation. Security is provided de facto by the LAN-specific basis of the client/server architecture.

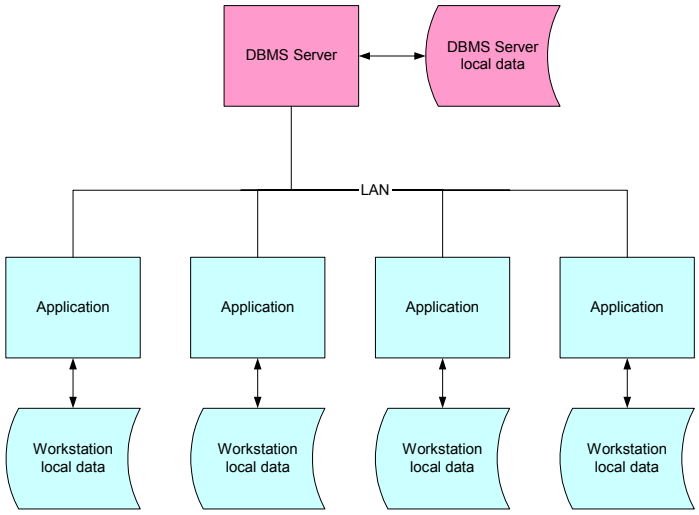

Figure 3 - Client/Server Architecture

Web CGI Forms

This architecture is an extension of client/server that employs a set of Internet/World-Wide Web protocols including HyperText Transfer Protocol (HTTP), (HyperText Markup Language (HTML), and Common Gateway Interface (CGI). Not originally intended for applications but for document sharing, the Web rapidly came to acquire application capabilities through the CGI protocol.

The client side utilizes a Web browser such as Netscape or MS Internet Explorer, accessing the server via HTTP as the top layer of a TCP/IP protocol stack. CGI programs are server-based executables that can accomplish any sort of task the server permits, including database access; however, no standard exists other than the presentation of forms on the client and the transport of forms data back to the server.

Little was new about this architecture over client/server (except, that is, for the revolutionary introduction of a simple way to process forms-based interactions over the Internet!). Much was lost in this major paradigm shift—notably security and standardized database access—but it lays the foundation for bigger and better architectures to come.

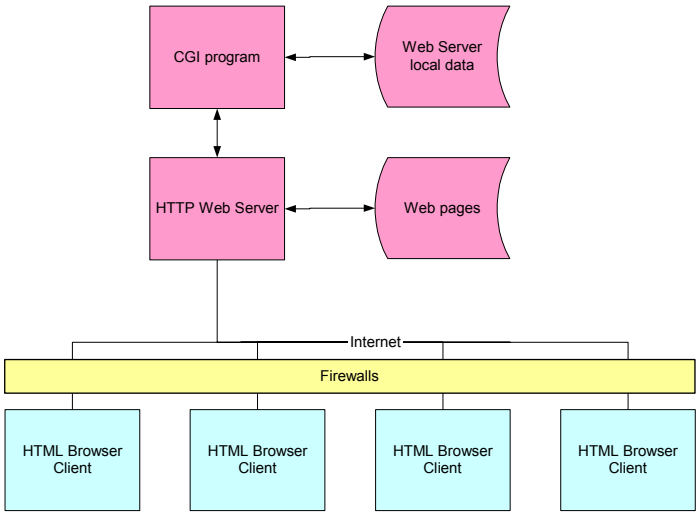

Figure 4 - Web CGI Forms

Intranet

Like client/server, this model is LAN-based, but the client is a "thin client" running in an HTML browser, as in the Web CGI Forms architecture. Virtually all business logic and data storage is located on the server side of the architecture; typically the browser's application role is limited to providing a forms-based GUI and doing limited field-level validation (using a scripting language like JavaScript) prior to form submission.

Intranets co-opted CGI for internal use within organizations; Extranets and ASPs carried the model back to the Internet with the addition of encryption and access control mechanisms, as we see in the next subsection.

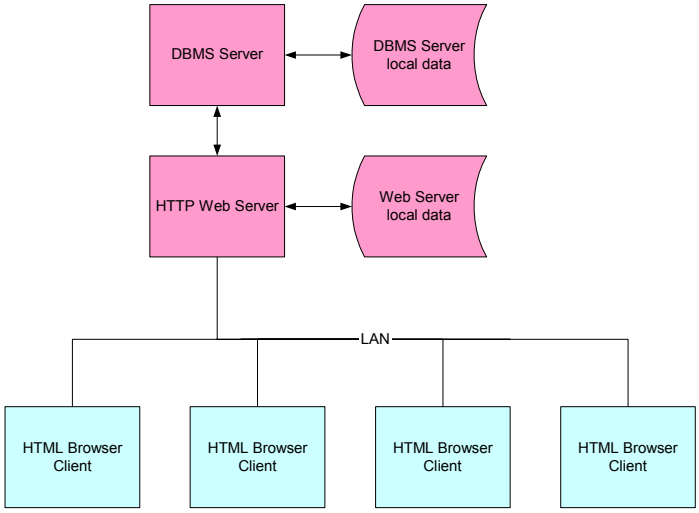

Figure 5 - Intranet Architecture

Extranet/ASP

This model is in many ways identical to the Intranet approach, but uses the Internet instead of a LAN. The secure equivalent of HTTP is used to provide encryption of packets going over the wire, and password protection is used to limit access to corporate resources. Clients can be anywhere on the Internet. Firewalls are often involved in the intervening layers of the network architecture, since the clients are often accessing the Internet from within a corporate LAN environment.

THE EXTRANET/ASP MODEL IS ESSENTIALLY IDENTICAL TO THE INTRANET APPROACH, BUT USES THE INTERNET INSTEAD OF A LAN.

This architecture is especially popular in supply chain applications and in conglomerate corporations resulting from the extensive M&A activities of the 1980's and '90's.

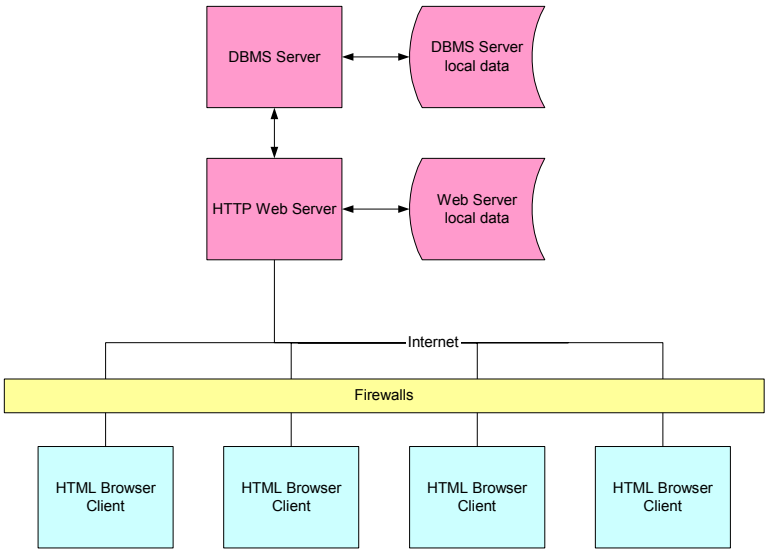

Figure 6 - Extranet Architecture

Application Server Architectures

Overview

This is a family of multi-tier architectures that are extensions of the Intranet or Extranet/ASP model, depending on whether the connectivity is LAN- or Internet-based. Application responsibilities are divided amongst three logical tiers in these architectures. More functionality may be distributed to the client, moving back in the client/server direction, but the application server architecture always involves the use of multiple servers to contribute different services to the application.

Distribution of functionality back to the client complicates application configuration management, but is an inevitable result of the growth in power on client platforms. Near-supercomputer power is a discount store commodity item in the early 21st century. A desktop configuration with an 800MHz processor, 128MB RAM, 20+ GB fixed disk, 56K modem, and 17" color monitor, fully multimedia-enabled and Internet-ready, can be purchased for well under $1,000 in any number of retail outlets. The most minimal Palm

Pilot configuration is more powerful in almost every way than the IBM PC of the early 1980's, and the higher-end Windows CE-powered Pocket PCs exceed the desktop computers available only a decade ago. A Linux-based handheld PC was released in March 2001 for under $300.

At least three conceptual layers are involved in the application server architecture. The client ideally handles details of the external interfaces of the system, especially the UI/GUI (model visualization) and any user interactivity. The second layer, the application server, is a control layer incorporating multiple client coordination (transaction management) and business logic. A persistent object store forms the third layer. Application server architectures are very flexible: any layer could conceivably be distributed across multiple machines, and any two layers could be combined on a single machine.

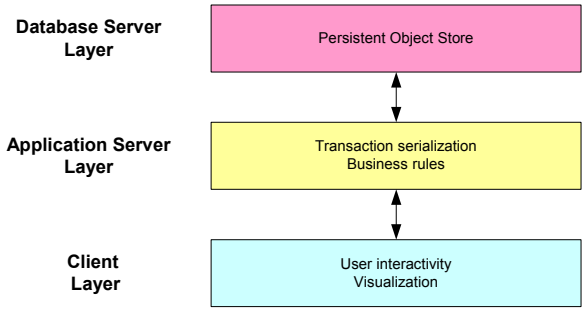

Figure 7 - Conceptual Layers In Application Server Architectures

Some examples of application server architectures include:

- Peer-to-peer
- Server-assisted peer-to-peer
- Collaboration Software Add-in
- Enterprise Application Framework Add-In
- Web Services

Peer-to-peer

In peer-to-peer architectures, each computer acts as both client and server, initiating and servicing requests as needed in the fulfillment of each application's goals. Moreover, each peer provides services to the others that conform to the roles played by application server and database server. Interactions in the peer-to-peer model are scripted conversations carried out through the exchange of datagrams. Protocols such as XML enable disparate hardware and OS configurations to carry on peer-to-peer conversations.

IN PEER-TO-PEER ARCHITECTURES, EACH COMPUTER ACTS AS BOTH CLIENT AND SERVER, INITIATING AND SERVICING REQUESTS AS NEEDED IN THE FULFILLMENT OF EACH APPLICATION'S GOALS.

In a peer-to-peer protocol, each transaction has a client/server appearance to it: the initiating party acts as client and the responding party acts as server.

What differentiates this protocol from client/server is that each party to the protocol has the capacity to initiate as well as respond to a transaction request.

Peers can interact with each other on a LAN without intervening firewalls, or on a WAN such as the Internet if a way is found to interact through any intervening firewalls. One common way of accomplishing this is called "HTTP tunneling," wherein requests are exchanged between peers in the guise of Web server request-response interactions. This is in theory only possible if the firewall allows each such client to act as both client and server on the HTTP port (typically port 80). While most clients have the ability to initiate HTTP requests, the ability to respond to HTTP requests is typically only given to server machines. This apparently insurmountable limitation can in practice be overcome using a server assist, as described below in the section entitled Server-assisted peer-to-peer.

Peer interactions are common between Internet servers, such as SMTP (mail) servers, but less common between clients in an application context. The file sharing service Gnutella is a recent example of a pure peer-to-peer application. This architecture often exhibits scalability problems and is best suited to small communities of peer applications.

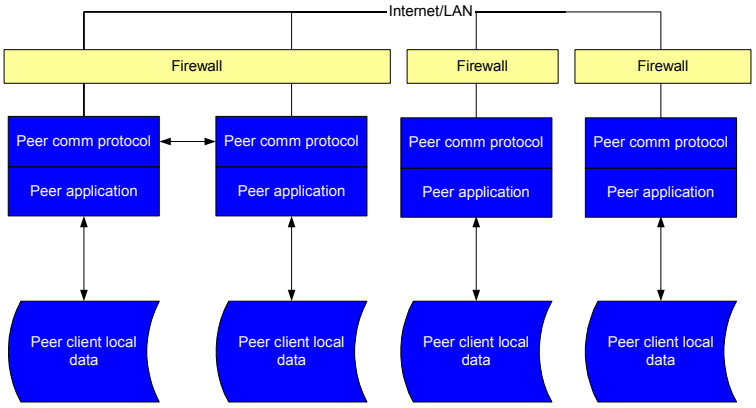

Figure 8 - Peer-To-Peer Architecture

Server-assisted peer-to-peer

Server-assisted peer-to-peer is a hybrid architecture. The clients use peer-to-peer interactions where appropriate in the application operation, but they rely on centralized servers to provide facilities that are better off centralized, such as directory and data base facilities. The server may also mediate interactions between clients behind firewalls by acting as middleman in an HTTP tunneling scheme.

The music file sharing service Napster is based on a server-assisted peer-to-peer architecture, successfully orchestrating the interactions of as many as 10,000 simultaneous users on a routine basis.

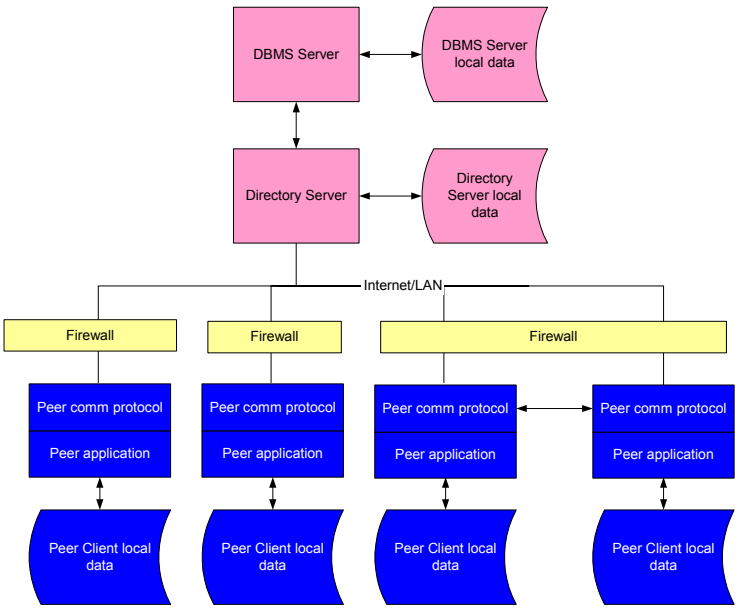

Figure 9 – Server-Assisted Peer-To-Peer Architecture

Collaboration software framework add-in

This architecture uses a general-purpose framework to support distribution of application functionality across servers and clients. The collaboration framework server typically fills the role of both application server and database server. Examples of collaboration software frameworks include IBM's Lotus Notes and Microsoft's Office front end with Exchange as the back end. Services such as electronic mail and persistent data storage may be built into the framework, but the developer must program business logic. In most cases, both fat and thin clients are supported.

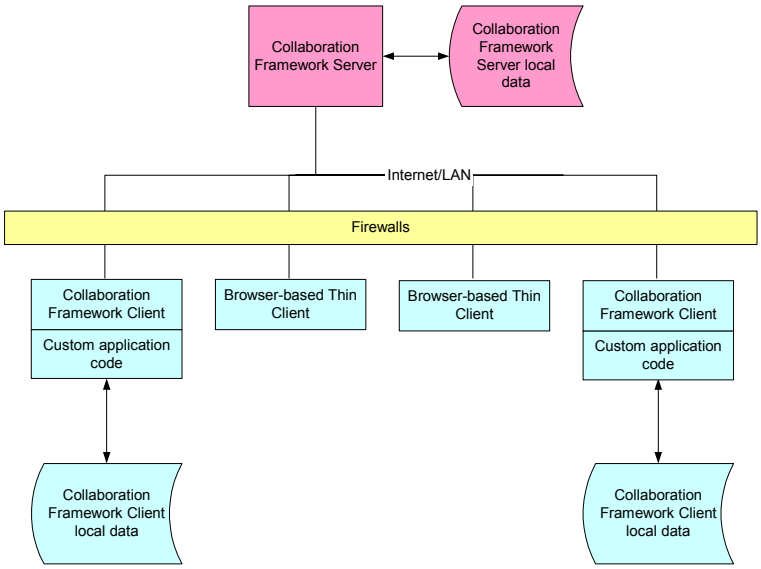

Figure 10 - Collaboration Framework Add-In Architecture

IBM's Lotus Notes, Microsoft's .NET initiative, and Oracle's e-commerce suite are examples of attempts to create all-encompassing collaboration software environments. On a more entrepreneurial note, Ray Ozzie, original architect of Notes, has recently released a peer-to-peer hybrid model collaboration software framework called Breeze. Breeze represents an entirely new approach to software architecture in the realm of office applications.

Enterprise Application Framework add-in (e.g., SAP, Oracle, PeopleSoft)

These frameworks extend the concept of collaboration software frameworks to include substantive "shrink-wrapped" implementation of business process logic. Third-party solutions partners associated with the enterprise framework vendors provide these kinds of products, which usually still require customization to the needs of the specific customer and application environment. In most cases, both fat and thin clients are supported.

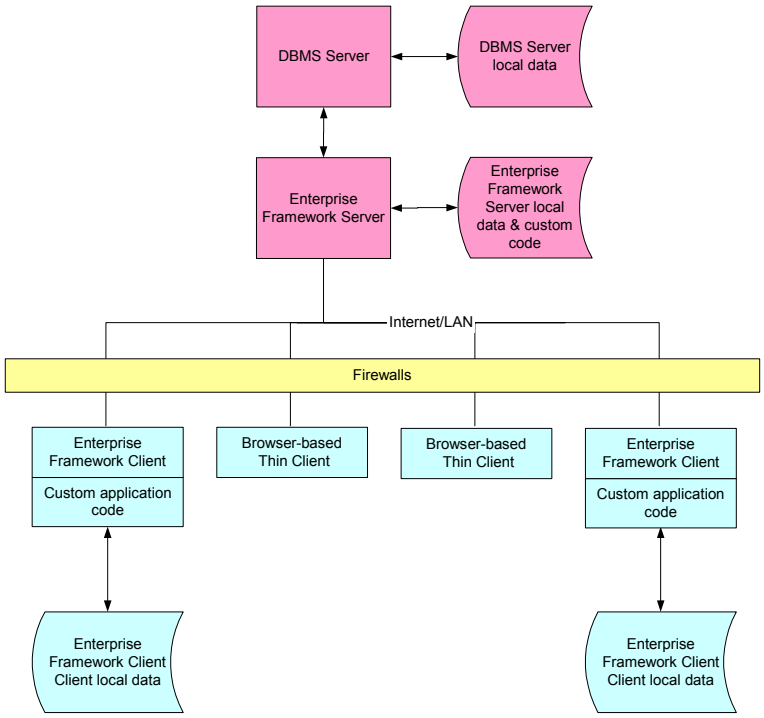

Figure 11 - Enterprise Application Framework Architecture

Web Services

Peer-to-peer hybrid architectures also exist in which the *servers* are peers. The most important example is the emerging Web Services[1] paradigm. This paradigm is expected

[1] The architecture we are calling Web Services is compelling enough that it will be adopted whether or not with its current name and protocol configuration. By way of example, Hewlett-Packard's E-Speak and Oracle9iAS Dynamic Services are Web Services under different names, though they will move under the

to dominate the applications marketplace over the next decade, as the client/server model dominated in the early to mid 1990's and the Intranet and Extranet/ASP models did the late 1990's and the early 2000's.

Architecturally, this form is not to be confused with the multi-tier client/server architecture, which consists of three or more layers in which each pair of adjacent layers relates in a predetermined client/server fashion. Instead, servers in a Web Services environment act as libraries of procedures that can be accessed by other servers, or directly by clients. Procedures are accessed via remote procedure calls.

THE BIG QUESTION IN THE WEB SERVICES REALM IS THIS: WHICH OF THE INTERNET-FOCUSED APPLICATION FRAMEWORKS WILL COME TO DOMINATE THE WEB SERVICES LANDSCAPE?

Offered services are described in a registry that can be accessed by any potential consumer of the services. The registry contains descriptions of the services offered (analogous to the telecom world's Yellow Pages), contact information for the service provider (analogous to White Pages), and calling protocols for each of the offered services (analogous to the Green Pages—the section in a telephone book that describe how the telecom service works).

The implication of the discovery and description mechanism is profound, and it remains to be seen how soon its full potential can be realized. With Web Services, it should be possible to allow computers to configure dynamically and automatically interactions that now require tedious negotiation of protocol definitions and manual programming.

Although there is no official standard for Web Services yet, the world seems to be settling on several standard protocols as their basis, as shown in Table 4 - Protocols Used In Web Services. These protocols allow highly automated electronic conversations that can implement an unlimited variety of interactions.

The big question in the Web Services realm is this: Which of the Internet-focused application frameworks will come to dominate the Web Services landscape? J2EE is ahead as of 1Q2001, with two players—BEA with its WebLogic software and IBM with WebSphere—sharing the limelight almost equally. Other J2EE implementations, especially Oracle9iAS and the Sun/Netscape alliance's iPlanet offering, are rapidly growing competitors for the two leaders. Microsoft, however, has put together a very compelling package in its beta versions of Microsoft.NET. .NET is an open standard, in spite of its origins at the world's most aggressive advocate of proprietary software. Expect Microsoft to provide a first-class implementation, but others may well join in the game if .NET begins to gain market acceptance.

ebXML versus UDDI: A Potential Barrier To Web Services Acceptance

There is a fundamental disparity of viewpoint within the Web Services community over the issue of standards comprehensiveness. The ebXML initiative attempted to define a set of standards that would cover all bases, from the service discovery through remote

---

Web Services banner when it is unfurled and the saluting starts. Furthermore, alternatives still exist for some protocols in the picture, but the abstract architecture will not be affected.

procedure calls to broad standards for marketplace transactions. This approach would provide an immediate basis for fully automated transactional systems, but requires a longer up-front investment with the attendant risk that proprietary initiatives would become de facto standards before ebXML was finalized. In contrast, the Microsoft/IBM/Ariba alliance that produced UDDI focused on getting the discovery, description, and remote procedure call interfaces standardized first, leading to a faster time-to-market but with the downside risk of creating a Babel of transaction-level standards.

Given the importance of the overarching goals, the two sides have made valiant attempts to resolve their differences, and an agreement reconciling the opposing camps is expected in 2Q2001.

Table 4 - Protocols Used In Web Services

Acronym	Protocol Name	Usage
HTTP	Hypertext Transfer Protocol	Transport of messages across firewalls
XML	eXtensible Markup Language	Syntax for all message content in SOAP, UDDI, and WSDL
SOAP	Simple Object Access Protocol	Semantics of remote procedure calls
UDDI[2]	Universal Discovery, Description, and Integration	Semantics of service discovery, description, and integration
WSDL	Web Services Description Language	Semantics of service descriptions (calling scenarios, call formats, call argument formats, response formats)

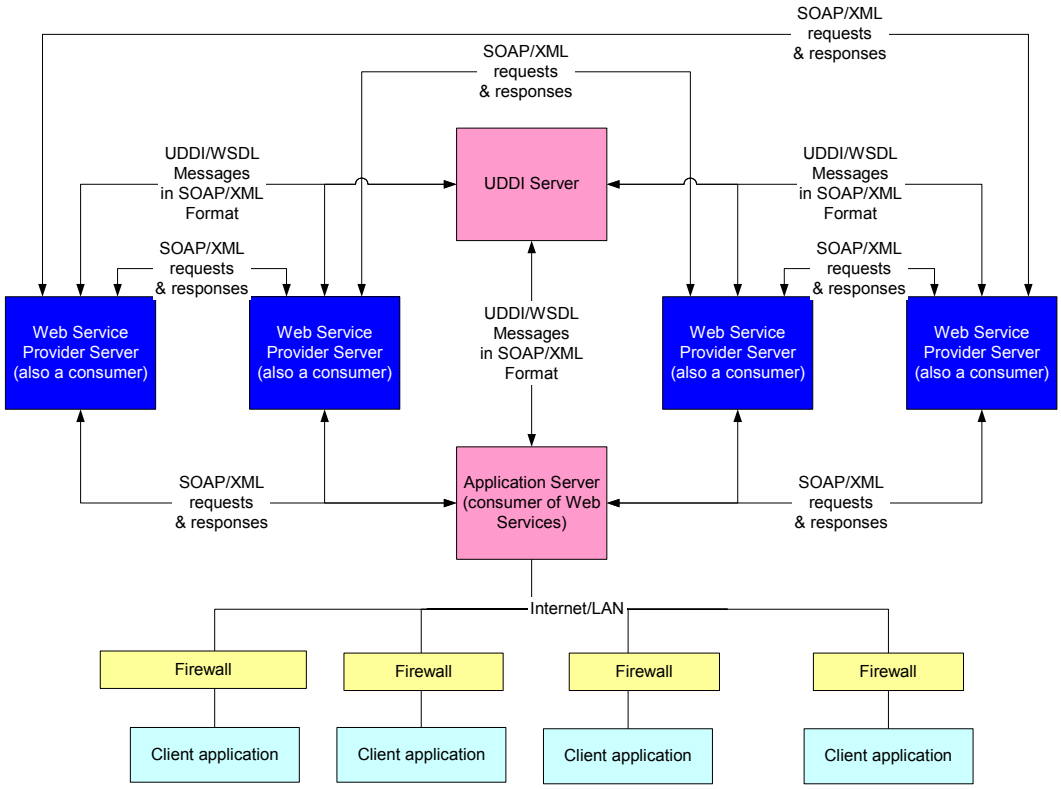

Figure 12 – Web Services Architecture

Additional features to watch

Multi-user applications in different problem domains seem to be converging on a set of features that are common regardless of the architecture in use. These features include:

- Device heterogeneity
- Intermittent connectivity

[2] UDDI and WSDL may be replaced or exist in competition with ebXML, an OASIS initiative.

- Workflow management
- Knowledge management

Device heterogeneity

The advent of wireless computing has produced a profusion of devices and device types that can act as application clients. Many applications will need to support access from multiple device types. The problem raised by multiple devices is not simply a formatting issue: different kinds of devices may require entirely different content selection and user interaction scenarios, due to limitations on screen size and bandwidth.

THE ADVENT OF WIRELESS COMPUTING HAS PRODUCED A PROFUSION OF DEVICES AND DEVICE TYPES THAT CAN ACT AS APPLICATION CLIENTS.

A three-tier architecture has been popular throughout the late 1990's and into the early 2000's, with the three layers being presentation, business logic, and storage management. Device heterogeneity adds an additional layer, responsible for device-specific filtering and translation, which is required to adapt the application to the user's device of choice. Figure 13 - Architectural impact of wireless connectivity illustrates this graphically.

Differing implementations of the adaptation layer usually follow the same basic approach. The business logic layer provides information to display in XML format. XSLT stylesheets are used to transform the XML to an appropriate format for the client device (e.g., WML for a WAP phone, a subset of HTML for a Palm Web Clipping application, or XHTML for a desktop or laptop running a Web browser).

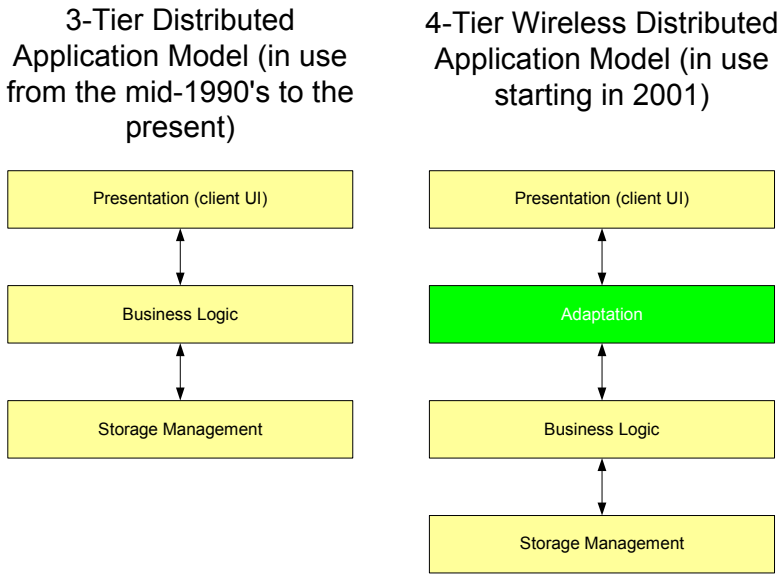

Figure 13 - Architectural impact of wireless connectivity

Intermittent connectivity: a vital feature for innovation

In the opinion of the writer, this architectural feature holds the most promise for groundbreaking applications in the early 21st Century. If intermittent connectivity is to be provided, the client must provide its own implementation of some subset of server functionality while disconnected, hence it requires some implementation effort. The subset typically includes data storage and retrieval and message queuing. A synchronization activity occurs whenever the intermittent connection is re-established. Complex reconciliation logic is required to support serializable database transactions in an intermittently connected architecture.

Intermittent connectivity architectures are a requirement for wireless applications, and immensely useful as a basis for telecommuting and remote workforce automation applications such as sales force and field service applications.

INTERMITTENT CONNECTIVITY REFERS TO THE INABILITY OF THE ENCOMPASSING SYSTEM TO GUARANTEE THE *POSSIBILITY* OF CONNECTION.

As used in this document, intermittent connectivity refers to the inability of the encompassing system to guarantee the *possibility* of connection, for example in the case of a laptop computer that must dial into an ISP to join a WAN, or a Web-enabled cell phone that may go in and out of the service area. However, intermittent connectivity architectures can be advantageous even where continuous connectivity is presumed to be available.

For example, in applications in which users are sporadically involved in heavily interactive sessions interspersed with periods of inactivity, intermittent connectivity can help maximize UI responsiveness, synchronizing during idle times. Moreover, the greater complexity of intermittent connectivity architectures can sometimes be justified simply because the cost of downtime is too great in a particular application. An ATM might be networked to its bank via a highly reliable well-shielded leased line connection, but in the relatively rare case that the WAN goes down, the ATM still must not lose any deposit transaction it has already accepted.

Furthermore, when considered in conjunction with device heterogeneity, we can see that intermittent connectivity transcends the assumption that the user will reconnect with the same client device as before—hence the state of the user's data is maintained on the network in a form that can be served to any device of the user's choice.

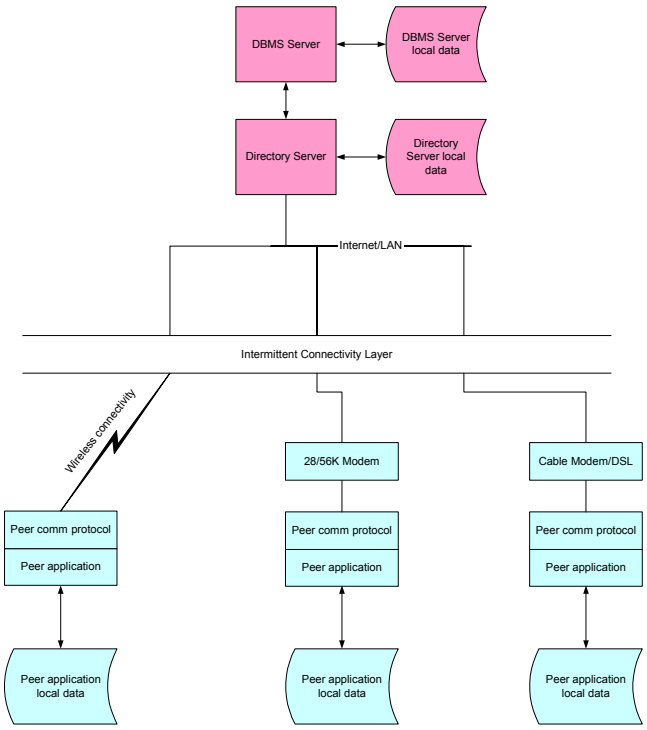

Figure 14 - A Server-Assisted Peer-to-Peer Intermittent Connectivity Architecture

Workflow management

Many enterprise applications can be modeled using a refinery approach. Information comes into the system; is normalized, refined, and/or transformed in some way (or perhaps one or more of several ways depending on the nature and timing of the inputs); undergoes a cycle of approval and aggregation; and is periodically released to the system's customers.

WORKFLOW MANAGEMENT USES QUEUING AND ROUTING TECHNIQUES TO TRACK AND CONTROL THE PASSAGE OF WORK ITEMS THROUGH THE SYSTEM.

This model obviously applies to information providers, such as news archiving and aggregation operations and the production of periodicals. However, it also applies to enterprises that manufacture complex products requiring extensive design and systems engineering activities, such as pharmaceuticals and aircraft. It applies to wholesalers and retailers planning inventory management and marketing activities based on seasonal cycles. It also applies to consultative sales organizations, where the closing of any given sale requires coordinated timing of efforts among several team members.

In any application of the refinery approach, workflow management is an important feature. Workflow management uses queuing and routing techniques to track and control the passage of work items through the system. The actors in the system are modeled as workstations, whose activities can be divided at an abstract level into scheduling, transformation, and quality control activities. Each workstation

type may actually represent multiple physical workstations, each of which has its own queue holding work waiting to be processed.

THE WORKFLOW IMPLEMENTATION MECHANISM CAN BE AN ADD-IN TO AN ELECTRONIC MAIL SYSTEM SUCH AS EXCHANGE OR NOTES, A CUSTOMIZABLE WORKFLOW MANAGEMENT PRODUCT, OR A COMPLEX CUSTOM DATABASE APPLICATION.

If multiple stations can accomplish the same task, assignment to a given workstation may be done according to a round-robin approach or by assignment to the station with the shortest queue. In the case of quality-control workstations, a given work item may require approval from any one of the QC workstations or from all such workstations; in the latter case, it may be possible to route work items to the QC workstations serially or in parallel, depending on the nature of the items.

Workflow management implementations range from the simple to the complex. The implementation mechanism can be an add-in to an electronic mail system such as Exchange or Notes, a customizable workflow management product, or a complex custom database application. Especially in the case of the distributed architectures described above, workflow management will be an increasingly common feature in the future.

At this point the Internet-focused frameworks do not offer a standardized way to do workflow. Oracle9iAS integrates workflow directly, as does the .NET platform with its BizTalk server.

Knowledge management

Many enterprise systems manage large quantities of complex content. RDBMS tools for searching such content have been given short shrift in the vendors' R&D budgets, with the emphasis instead put into such areas as performance optimization, data warehousing, and analytical processing.

Enterprise managers' needs and interests are shifting in a direction that some practitioners have begun to call "knowledge management." The inclusion of the word "knowledge" in this term threatens to open up endless epistemological debates, but the underlying expectation is clear—just as a CEO expects the data processing systems to "do what I mean," a search against the enterprise's content base is expected to "find what I mean."

Such content has been partitioned between paper-based file systems and rigid, simplistic database schemata for the last fifty years, so the "find what I mean" issue has been academic or laughable, depending on your point of view. However, recent advances in document processing have made this goal much more achievable. These advances include:

- Invention of semantic markup languages such as SGML and XML
- Advances in optical character recognition, handwriting recognition, and speech recognition that allow capture of information not originally stored as computer-accessible text

- Advances in computational linguistics that allow the extraction of lexicons from bodies of content
- Development of techniques such as concept-based search and fuzzy search that allow more sophisticated targeting of natural-language queries

THE KNOWLEDGE LAYER SUPPORTS SOPHISTICATED QUERIES AGAINST HETEROGENEOUS CONTENT STORES.

Taken together, all these techniques can be said to enable the creation of a "knowledge layer overlaying the entire body of enterprise content, a layer that represents the structure of the body of knowledge that constitutes the primary intellectual asset of the enterprise (See Figure 15 - Separation of Knowledge and Content). This knowledge layer supports sophisticated queries against heterogeneous content stores, which may include databases, microfilm or microfiche stores, image repositories, and even paper file systems if they have been adequately catalogued. It presents new opportunities as well as new management challenges that must be addressed by emerging application architectures.

Toolsets for dealing with this challenge are emerging from a number of startup ventures, and we expect they will be acquired and integrated into RDBMS as they evolve over the next few years.

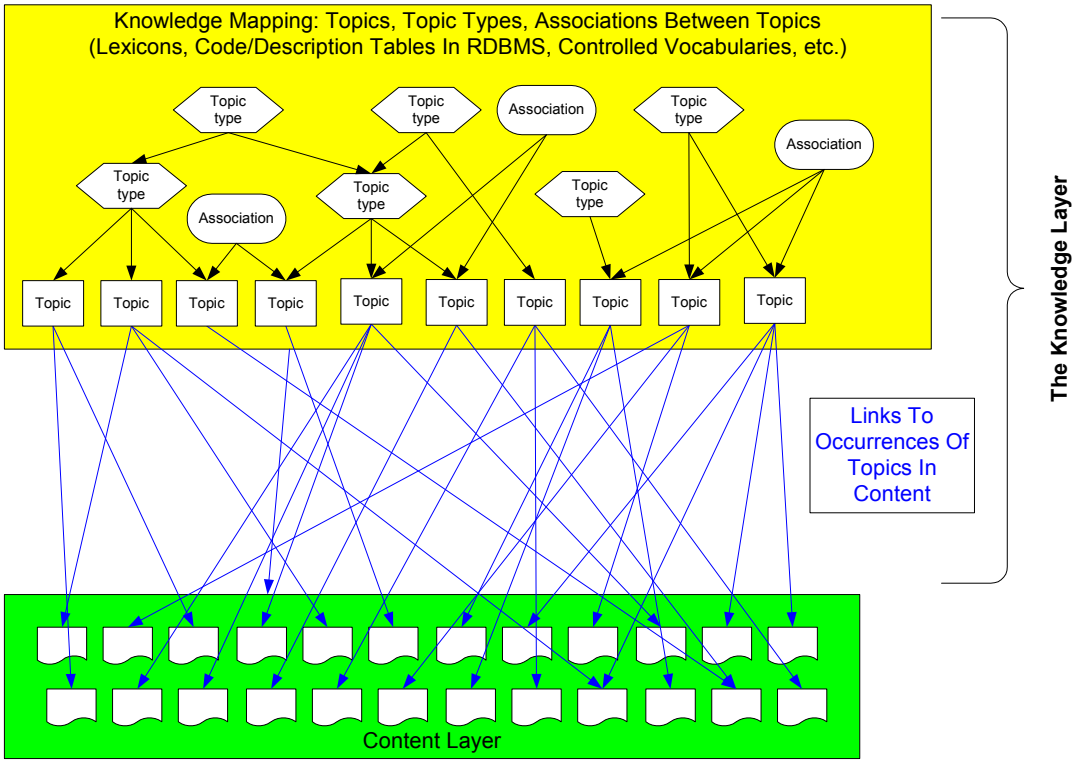

Figure 15 - Separation of Knowledge and Content

Summary: The Major Application Architectures In 2001

The following table summarizes the features and functions of the various application architectures.

Table 5 - Application Architectures Features & Functions Matrix

Application Architecture	Client	Server	Connectivity	Firewall	DBMS	Messaging	Market Trend
Workstation - standalone	Fat	None	None	None	Local or none	None	Large, declining rapidly
Client/Server	Fat	Single-tier	Continuous, LAN	None	Server-based	synchronous IPC	Large, declining slowly
Intranet	Thin	N-tier	Continuous, TCP/IP	None	Separate server	HTTP (plus others?)	Growing inside orgs
Extranet/ASP	Thin	N-tier	Intermittent, TCP/IP	Yes	Separate server	HTTP (plus others?)	Extranets growing inside orgs; ASPs stumbling a bit
Distributed application	Multiple types	N-tier	Continuous or intermittent, LAN or TCP/IP	Yes	Separate server	HTTP (plus others?)	Small, growing
Peer-to-peer	Acts as both at all times	Acts as both at all times	Continuous or intermittent, LAN or TCP/IP	Likely	Distributed amongst peers	Proprietary; BEEP	Small; future uncertain
Server-assisted peer-to-peer	Acts as both once connected to peer	Server used for directory purposes	Continuous or intermittent, LAN or TCP/IP	Likely	Located on server	Proprietary; BEEP	Large early adopter in place; future uncertain but more positive than simple peer-to-peer
Application Server Architectures:							
Collaboration Software Add-in (e.g., Notes, Exchange)	Fat	Possibly N-tier	Continuous or intermittent, Provided by framework	Likely	Provided by CS	Provided by CS	Large, stable
Comprehensive Framework Add-in (e.g., SAP, Oracle, PeopleSoft)	Fat	N-tier	Continuous or intermittent, Provided by framework	Likely	Access provided by framework	Provided by framework	Large, stable
Web Services	Typically thin	Multiple servers acting in server-assisted peer-to-peer mode	Continuous or intermittent, LAN or TCP/IP (servers continuous, clients intermittent or continuous)	Possibly multiple firewalls	Located on one or more servers	HTTP (plus others?)	Small, but expected to start growing very quickly due to widespread industry support

Future Trends In Application Architectures

To develop a picture of future trends, we try to partition relevant influences into predetermined elements and critical uncertainties. Then by looking at potential permutations of uncertainty outcomes we can derive some alternative scenarios regarding our subject of interest.

Predetermined Elements

Several factors seem to be shaping up as predetermined components of any future scenario. Among these are the following:

- Continuing RDBMS Dominance
- Intermittent Connectivity Prevails
- Ubiquity of Internet Access
- Increasing Popularity of Wireless Connectivity
- Reactive nature of the Open Source Movement

Continuing RDBMS Dominance

When the trends in the commercial arena are coupled with the slow but steady maturation of the Open Source movement's offerings, it seems reasonable to assume that commercial RDBMS market dominance will continue at the high end of the market, with some penetration by Open Source alternatives at the low to medium end of the spectrum.

Intermittent Connectivity Workflow, and Knowledge Management

Distributed applications will rule this century, and intermittent connectivity architectures will define the decade beginning in the current year, 2001. While not all new products and service delivery mechanisms will require this type of architecture, technical specifications will need to justify why the product or service they describe is *not* using intermittent connectivity, rather than why it *is* using such architecture.

Likewise, workflow and knowledge management features will be common features of new system architectures in the next few years. Not all systems will require them, but they will begin to appear in new applications addressing many problem domains.

Ubiquity of Internet Access

Ten years ago, access to the Internet from corporate desktops was a rarity. During the 1990's, however, first electronic mail and then World-Wide Web access became standard features of the corporate desktop. Access from remote locations such as homes, airports, and locations is expected to become a certainty within the next three to five years.

Increasing Popularity of Wireless Connectivity

It is already possible to access the Internet via a wireless connection using second-generation cellular networks (2G, or digital cellular service). The Palm VII is an example of such a device; several hundred thousand of these are in use as of early 2001. It seems certain that access to the Internet via PDAs, Web-enabled cellular telephones, and other mobile devices will continue to grow in popularity and availability. Two forms of wireless connectivity are already taking off: wireless LAN technology and packetized digital cellular transmission.

Wireless LAN technology, based both on the existing IEEE 802.11 (Wi-Fi) Wireless LAN standards family and the emerging IEEE 802.15 (Bluetooth) standards family, is expected to become universally available within corporate environments and urban public spaces. Wi-Fi connectivity is more expensive and can reach bandwidths up to about 10 megabits per second (MBps) already. Bluetooth is far less expensive, targeted at low-end consumer devices and appliances, but will still achieve bandwidth levels up to 2MBps, albeit only within a relatively short range of a transmitter (approximately 30 meters).

WIRELESS LAN TECHNOLOGY IS EXPECTED TO BECOME UNIVERSALLY AVAILABLE WITHIN CORPORATE ENVIRONMENTS AND URBAN PUBLIC SPACES.

Third-generation (3G) cellular service is expected to roll out in Europe and Asia in the next two years and in America by 2005 with bandwidth in the 2MBps range, approximately equivalent in bandwidth to 802.11b. 2.5G cellular service (packet-switched service over existing digital networks), available today in some locations worldwide, can reach 40 kilobits per second (KBps) bandwidth or better (2G service is limited to about 14.4KBps). By the end of the year, 2.5G service from Sprint and Cingular is expected to reach 144KBps rates.

The rollout of *high-bandwidth* connectivity is not a certainty, however. Political and economic barriers to implementation exist in many locales; in particular, a global recession could easily diminish the corporate and governmental willpower required to achieve a global high-speed network. Services that operate over 2G networks (such as the existing digital networks that now cover most of the United States, Europe, and the more developed countries in Asia) will be the minimum available service level for wireless connectivity.

Regardless of the ultimate bandwidth available, though, device heterogeneity will be the rule rather than the exception in the future. Moreover, users will rightly expect that each device's view of the data will jibe with every other device and that each device will perform to its potential, so the issue is not simply one of adjusting presentation. Because many of these devices will have substantial computing and storage on the device itself, users will expect the device to operate in spite of connectivity, so data synchronization will be an ongoing issue.

Reactive Nature of the Open Source Movement

The Open Source Movement has been characterized as a quixotic and idealistic effort at times, but in actual fact it is really a grassroots extension of the demand for universal standards in the software arena. Hardware manufacturers have long known the

advantages of standardized components, but proprietary players have thus far mostly dominated the world of software. Open Source proponents have now successfully established a freely available and commercially viable OS and full-featured Web server (Linux and Apache respectively).

Open Source software has thus far been developed reactively. Once commercial players establish the viability of a market and begin to exact the toll they require to please their stockholders and feed their R&D efforts, Open Source activities develop that attempt to provide free replacements for commercial components. Only in the arena of low-level software tools such as XML libraries and languages like Perl and Python does one see a glimmering of creativity and proactive thinking. Moreover, Open Source efforts seem unable to "get it" when it comes to user productivity software; existing Open Source office application suites and application frameworks are generally cumbersome and inadequate, and enterprise application framework alternatives don't even appear to be on the drawing board yet.

IF THE OPEN SOURCE MOVEMENT WERE TO LOSE ITS "GRUNT CODER" MINDSET AND ITS CONSEQUENT REACTIVE ORIENTATION AND BEGIN TO ATTACK HIGHER-ORDER PROBLEMS WITH THE SAME ZEAL THAT LED TO LINUX AND APACHE, GREAT THINGS ARE POSSIBLE.

This situation need not continue forever, though. If the Open Source Movement were to lose its "grunt coder" mindset and its consequent reactive orientation and begin to attack higher-order problems with the same zeal that led to Linux and Apache, great things are possible. Such a turn of events would bode well for applications-oriented product and service companies, and bode ill for tools-oriented product companies.

Critical Uncertainties

Two factors seem to be shaping up as critical uncertainties in future application architecture scenarios.

- Bandwidth of wireless connectivity
- Prevalence of Open Source versus Microsoft in the Internet-focused application frameworks arena

Bandwidth of Wireless Connectivity

While wireless access is expected to become ubiquitous in some form or other, it is uncertain whether the technological and political barriers to ubiquitous high-bandwidth access will be solved in the near future. If IEEE 802.11b or some similar standard were to become the basis for ubiquitous wireless Internet access, data rates of 1 to 2Mbps will become the norm, equivalent to a typical cable modem environment today. However, such a high-bandwidth connectivity environment may not readily emerge, for any number of reasons, including politics, expense, or technological barriers.

Even in such an event, the less expensive Bluetooth standard is still expected to grow dramatically alongside the existing cellular network. In this networking environment,

much slower data transfer rates are the rule, equivalent to a 9600-baud modem, a performance level well below the normal telephone ISP hookup today. This would imply that wireless access would be limited to less bandwidth-intensive applications.

It must be noted that if a high-bandwidth network becomes a reality, both bandwidth levels will prevail—it is difficult to imagine a scenario in which existing cellular access and the emerging Bluetooth connectivity will not grow and prosper.

Java 2 Enterprise Edition Versus Microsoft

The other critical uncertainty exists in the arena of Internet-focused application frameworks. The battle in the Internet-focused application frameworks arena comes down to a battle between the J2EE platform and the .NET initiative.

Both .NET and J2EE are far more than programming languages or IDEs. Rather they represent mutually exclusive visions of the future paradigm for computing, paradigms that offer different routes to very similar visions of pervasive computing environments, where devices and applications will disappear from view and become part of the user's environment.

Both .NET and J2EE are fundamentally open standards, but the first implementations of tools for each environment express the fundamental differences in world-view. The Microsoft.NET beta may be based on open standards, but Visual Studio.NET represents an unmistakably slippery slope leading back to the seductive world of Microsoft operating environments. Conversely, the implementations of J2EE-compliant environments treat Windows machines as just another target for server or client applications, tempting developers to certify their applications for diverse client and server environments.

ALTHOUGH THEY ARE EXPECTED TO BE MUTUALLY EXCLUSIVE WITH RESPECT TO TOOLS AND IMPLEMENTATIONS, J2EE AND .NET ARE FULLY INTEROPERABLE.

Although they are expected to be mutually exclusive with respect to tools and implementations, J2EE and .NET are fully interoperable, thanks to early agreements on key factors such as the Web Services protocols (XML, UDDI, SOAP, WSDL, and HTTP). Consequently, organizations can pursue both paths in pilot projects without making a final commitment for some time to come. The marketplace choice between J2EE and .NET will come down to such factors as quality of tools, customer confidence in the vendors, and product pricing and packaging options.

Java has garnered significant support from a wide variety of backers and is the *langue du jour* amongst today's programmers. With the emergence of Java 2 Enterprise Edition, it has become not just a programming language but a complete framework for the continuous- and intermittent connectivity applications server architecture we expect to predominate in new systems development for at least the next decade. Consequently, this option is a very strong contender for dominance in the emerging world order.

Moreover, the antitrust suit has drained Microsoft's energies, while at the same time its product development groups appear to be without strong direction, just marking time.

There seems to be cause to believe that Microsoft will go the way of Borland and Ashton-Tate, albeit more slowly and with the lumbering gait of an NCR or Unisys.

However, Microsoft has been prematurely written off in the past on numerous occasions and has proven its critics wrong every time. Remember that: *every time*. It seems safe to assume the battle between J2EE and Microsoft .NET for Internet developer mind share is a critical uncertainty.

Future Scenarios

THE THREE SCENARIOS IN NO WAY REPRESENT THE SORT OF "HIGH ROAD/LOW ROAD/MIDDLE OF THE ROAD" BREAKDOWN THAT IS OFTEN FOUND IN TRADITIONAL FORECASTING.

To envision some possible future scenarios for application architectures, we plot a matrix showing the permutations of our critical uncertainties. This matrix is shown in Table 6 – Matrix Of Critical Uncertainties.

Readers should note that in the event of failure to attain the goal of ubiquitous high-bandwidth connectivity, it makes less difference which side wins in the server market, because the variety of application architecture flavors will be limited whether .NET or J2EE prevails. For this reason only one low-bandwidth scenario is presented.

Another important point is that the three scenarios in no way represent any sort of "high road/low road/middle of the road" breakdown as is often found in traditional forecasting. The three scenarios attempt to envision three distinct alternative futures that could result from the interplay of predetermined elements and critical uncertainties that now exist.

Table 6 – Matrix Of Critical Uncertainties

Internet-Focused Framework	Ubiquitous Bandwidth Level: High	Ubiquitous Bandwidth Level: Low
Microsoft .NET	**Windows All Around Us** • Microsoft wins anti-trust lawsuit • Microsoft's .NET vision prevails • High- and low-bandwidth connectivity ubiquitous • Near-term good news for users	**Windows Without A View** • Political or technological issues derail prospects for ubiquitous high-bandwidth connectivity away from the corporate desktop • Microsoft wins marketplace due to inertia in desktop and server markets • Either Microsoft wins anti-trust lawsuit, or the server market stagnates • Hemorrhage of server market share stabilizes, with Linux holding a significant minority • Cellular, Palm, and Bluetooth collectively and de facto define the intermittent connectivity foundation • State-of-the-art applications focus on device interconnection
Java 2 Enterprise Edition	**Caffeine Nation** • Microsoft loses anti-trust lawsuit, or gets lost in the "gray cloud" of uncertainty • J2EE and Linux prevail • Windows clients still abound • High- and low-bandwidth connectivity ubiquitous • Bad news initially for users as Microsoft's support deteriorates along with its profits • Better news for users later as greater creativity of an Open Source marketplace is unleashed	

Each of the three scenarios is described below.

Windows All Around Us

In this scenario, Microsoft wins the anti-trust lawsuit, and its .NET vision prevails in the marketplace for server software. High- and low-bandwidth connectivity becomes ubiquitous inside and outside corporate boundaries, thanks to as-yet-unforeseen favorable events in the political and technological arenas.

The Windows All Around Us scenario holds much near-term good news for ordinary home and corporate users. The desktop environment and applications they have come to

know and love over the past few years will remain in place, evolving into bigger and better .NET-enabled versions of themselves.

Cardinal signs of Windows All Around Us emergence

- Microsoft prevails in anti-trust lawsuit
- High-bandwidth technology rolls out rapidly and smoothly, possibly due to as-yet-unannounced joint industry-government initiatives

THE WINDOWS ALL AROUND US SCENARIO HOLDS MUCH NEAR-TERM GOOD NEWS FOR ORDINARY HOME AND CORPORATE USERS.

Good business bets in case of Windows All Around Us emergence

- Add-in applications that leverage .NET applications like the next-generation Internet-enabled Microsoft Office and Exchange packages
- Consulting services to assist in corporate migration to .NET and integration of wireless LAN/WAN technologies
- Pre-sale installation services to perform home network setups in new homes at all price levels, focusing on IEEE 802.11 or similar standard

Caffeine Nation

In this scenario, Microsoft loses its anti-trust lawsuit and is forced to split into two corporations; the two corporations fail to find ways to work together. This scenario also evolves if there are endless appeals and Microsoft continues to exist under the gray cloud that has paralyzed it for two years now.

CAFFEINE NATION HOLDS GOOD NEWS FOR USERS IN THE LONG RUN AS THE GREATER CREATIVITY OF AN OPEN SOURCE MARKETPLACE IS UNLEASHED AND NEW UI PARADIGMS AND APPLICATION ARCHITECTURES ARE EXPLORED.

J2EE and Linux prevail in the server marketplace, though Microsoft continues to maintain significant server market share and Windows clients still abound. As in the preceding scenario, high- and low-bandwidth connectivity become ubiquitous inside and outside corporate boundaries.

This scenario is bad news initially for end users as Microsoft's support deteriorates along with its profits. There will be better news for users later as the greater creativity of an Open Source marketplace is unleashed and new UI paradigms and application architectures are explored.

Cardinal signs of Caffeine Nation emergence

- Microsoft loses in anti-trust lawsuit, or loses direction while living under "gray cloud"

- High-bandwidth technology rolls out rapidly and smoothly, possibly due to as-yet-unannounced joint industry-government initiatives
- Open Source and J2EE movements continue to gain momentum and market share

Good business bets in case of Caffeine Nation emergence

- High value-added enterprise framework applications that live on top of Open Source foundations and leverage the new world of ubiquitous high bandwidth, leapfrogging the client/server-oriented existing market leaders
- Multi-targeted GUI frameworks that can handle a variety of display form factors well
- Agent technologies that leverage the ubiquitous connectivity in creative new ways

Windows Without A View

In this worst-case scenario, Microsoft wins the overall marketplace due to inertia in desktop market. Political or technological issues derail prospects for ubiquitous high-bandwidth connectivity away from the corporate desktop, leaving users largely still chained to their cubicles. The hemorrhage of Microsoft's server market share stabilizes, leaving Microsoft still in charge but with Linux holding a significant and very stable minority share.

IT IS FUNDAMENTALLY LESS EXCITING FOR YOUR REFRIGERATOR TO TALK TO YOUR WATER HEATER THAN FOR YOU TO WATCH JULIA ROBERTS AND BRAD PITT FALL IN LOVE ON YOUR POCKET PC AS YOU SIP YOUR PIÑA COLADA ON A BEACH IN BELIZE!

Cellular, Palm, and Bluetooth form the foundation of intermittent connectivity applications. There is much innovation in this arena, but it has little glitz because it focuses on device interconnection applications.

This scenario is good near-term news for users since they once again get to keep their familiar Windows desktop software. But this world is less interesting at a certain level than Windows All Around Us: it is fundamentally less exciting for your refrigerator to talk to your water heater than for you to watch Julia Roberts and Brad Pitt fall in love on your Pocket PC as you sip your piña colada on a beach in Belize!

Cardinal signs of Windows Without A View emergence

- Microsoft prevails in anti-trust lawsuit, or at least begins to show signs of recovering its sense of direction and purpose
- High-bandwidth technology support fails to coalesce

Good business bets in case of Windows Without A View emergence

- Add-in applications that leverage .NET applications like the next-generation Internet-enabled Microsoft Office and Exchange packages

- Consulting services to assist in corporate migration to .NET and integration of wireless LAN/WAN technologies
- Pre-sale installation services to perform home network setups in new homes at all price levels, this time focusing on Bluetooth as the core technology
- Tools and services to ease migration of client/server applications to an intermittent-connectivity model

Software Architecture Implications Of Future Scenarios

In all scenarios, we expect the low-bandwidth Bluetooth initiative to achieve significant penetration both in corporate and consumer markets. Peer-to-peer application architectures (both pure and hybrid) will predominate in the Bluetooth space.

The first axis of variation is the bandwidth issue. Can the developed world roll out a ubiquitous high-bandwidth wireless network over the next three to five years, or will technological or political issues get in the way?

If ubiquitous high bandwidth for wireless applications fails to become a stable part of urban culture across the developed world in the next few years (the Windows Without A View scenario), it is difficult to envision significant change in the balances of power. Heavy investments have already been made in Intel-based workstation and server hardware and the Microsoft Windows family of operating systems. If the promise of pervasive computing fails to materialize, the majority of large-scale enterprises will likely to continue to consolidate productivity gains already underway with existing investments rather than undertake risky new investments.

CAN WE ROLL OUT A UBIQUITOUS, HIGH-BANDWIDTH WIRELESS NETWORK OVER THE NEXT THREE TO FIVE YEARS, OR WILL TECHNOLOGICAL OR POLITICAL ISSUES GET IN THE WAY?

Server-based continuous-connectivity architectures (client/server, Intranet, and Extranet) will prevail in this scenario. Innovative applications will still arise, but they will focus on finding better solutions to existing problems rather than creating entirely new market categories. The sole exception will be in the low-bandwidth Bluetooth market space.

If ubiquitous high bandwidth wireless connectivity is achieved, the second axis of variation comes into play. This dimension is the choice between the competing overarching visions of the future of distributed computing, Microsoft .NET (leading to the Windows All Around Us scenario) and Java 2 Enterprise Edition (leading to the Caffeine Nation scenario). In either case, most significant applications will be distributed and will involve some peer-to-peer interaction. We also expect that most will employ intermittent connectivity as an architectural feature.

In either scenario, we expect the face of computing to change dramatically over the next five years. Entirely new market categories could easily emerge, and the desktop paradigm that defined the first twenty years of personal computing will to some degree be overshadowed by the integration of computing power into everyday appliances, furniture, and even clothing.

If the Microsoft vision prevails, the Windows-centricity and collaborative application framework of the excellent Microsoft toolset will largely predetermine application features and functions. But this Windows-centricity is not inherent in the .NET paradigm itself, which is platform-neutral and open, in spite of the Open Source community's suspicions. A Microsoft victory would have the effect of delaying the decline of desktop computing somewhat. Contrariwise, if J2EE prevails, its greater degree of OS independence will mean more breathing space for Linux and possibly a revival of the Macintosh OS, along with a nearly immediate proliferation of new device and interface types.

WHICHEVER VISION (J2EE OR MICROSOFT.NET) PREVAILS, THE OTHER WILL STILL BE A SIGNIFICANT "OPPOSITION PARTY" WITH FERVENT ADHERENTS AND SUPPORT.

Whichever vision prevails, the other will still be a significant "opposition party" with fervent adherents and support. In the Windows All Around Us scenario, Java adherents will continue to grow their vision and will gain adherents, mostly as a share of the growth of the overall market rather than conversion of existing sites. Conversely, in the Caffeine Nation scenario, inertia in the Windows installed base will leave Microsoft a significant presence for the foreseeable future, following a course analogous to Netware's slow decline.

Preferred Applications Architecture(s) in the Windows All Around Us scenario

- Application features and functions are largely predetermined by the Windows-centricity and collaborative application framework of the .NET paradigm
- Collaborative framework add-in architectures abound
- Client/Server, Intranet, and Extranet architectures decline in market share slowly, and their proponents continue to be prosperous for the foreseeable future
- J2EE is still a significant "opposition party" using a variety of distributed architectures
- As with each of the other scenarios, peer-to-peer applications will predominate in the Bluetooth space

Preferred Applications Architecture(s) in the Caffeine Nation

- Most significant applications will employ continuous or intermittent connectivity architectures whose features and functions will be shaped by the J2EE paradigm
- Microsoft will still have a strong presence, given that its Windows installed base is not going to go away anytime soon; new Windows-centric applications will be based on the collaborative application framework of the .NET paradigm
- As with each of the other scenarios, peer-to-peer applications will predominate in the Bluetooth space

Preferred Applications Architecture(s) in the Windows Without A View scenario

- Most significant applications will employ continuous-connectivity architectures
- Early adopters will split about evenly between .NET and J2EE, but will primarily use less expensive continuous-connectivity architectures
- As with each of the other scenarios, peer-to-peer applications will predominate in the Bluetooth space

Reading The Tea Leaves

So in a practical sense, what are the business implications of all this for software architects, entrepreneurs, and investors?

We see three important points each of us should keep in mind as we consider the future:

- **Watch the economy carefully**
- **In an economic recovery scenario, Web Services will rule**
- **Hedge your bets**

Watch The Economy

Keep an eye on present and future indicators. At South Wind, we watch the Nasdaq and Dow indices, with a special eye on technology stocks like the following:

- Microsoft (MSFT)
- Sun Microsystems (SUNW)
- Oracle (ORCL)
- Intel (INTC)
- BEA (BEAS)

Along with those markets, we watch the leading economic indicators such as consumer confidence, manufacturing spending, etc. to get an insight into the future. We also watch the world economy carefully, with a keen eye turned toward the Japanese and European markets.

Although Chairman Greenspan is a minor deity in our eyes, we consider the Fed's rate cuts to be fundamentally irrelevant to the technology world—they take effect too slowly to have much bearing on our recommendations. We pay a lot more attention to new VC investments, getting our information through Web sites such as http://www.ipo.com with its DealWatch area.

In An Economic Recovery, Web Services Rule

In each of the recovery scenarios (Windows All Around Us and Caffeine Nation), Web Services plays an increasingly important role over time, as the World-Wide Web grows

in importance as a financial transaction medium. Web Services-related investments should pay off handsomely over the next few years if the economy can support renewed expansion of the New Economy. A couple of factors need to fall into place for Web Services to reach its full potential, notably the establishment of a global services directory and the coexistence or merger of the ebXML and Web Services movements.

The prospects for Web Services are less rosy over the next few years if the economy remains in the doldrums. The technology will eventually see its day, but that day will not be soon if the world business community remains hunkered down and fearful of infrastructure investments.

Hedge Your Bets

There are a few points to consider that will serve you well whichever way the world goes.

First, nurture your relationships with existing customers. While this is obviously a Good Thing, when new customer sales are strong it can be easy to forget how important your installed base really is. When the economy stays sour for any length of time, though, your existing customers may be your best source of revenue.

Second, hone your legacy skills. In old-fashioned terms, make sure you stick to your knitting even if you have one eye on the bleeding edge.

And third, do keep that eye on the bleeding edge. Invest as much time and energy as you can in coming up to speed on the new technologies, even if it's a stretch. When times are tough, it can be difficult to follow through on this, but when the sun comes out again you will be the first one on the beach, ready to surf the next wave. It's a heady feeling when that time comes!

Terminology And Abbreviations

Term	Meaning as used in this document
.NET Initiative	Microsoft's multi-year blueprint for transitioning to a pervasive computing environment.
2.5G, 3G	These refer to different generations of wireless data transmission service. First-generation (1G) wireless service is analog, and capable of reliable data transmission at speeds of 9.6 KBps or less. Second-generation (2G) wireless is digital, and capable of data transmission at speeds up to about 14.4 KBps. 3G service is defined by the FCC as 144 KBps in a vehicle, 384KBps while walking, and 2MBps or higher indoors; 3G service is not currently available in the US due to a requirement for changes to the cellular infrastructure. 2.5G service is not officially defined, but usually refers to data transmissions at speeds from 28 to 144 KBps, which can be accomplished by more efficient use of existing infrastructure. Sprint and Cingular are both expected to roll out 2.5G services within calendar 2001; ubiquitous 2.5G should be available in the US by the end of 2002, and 3G by early 2004. Europe and Japan will roll out 2.5G and 3G services much sooner.
3GL, 4GL	Third-generation and fourth-generation computer languages.
Agents	Application programs that operate in the background without human intervention in the service of predefined goals.
ASP	Application Service Provider; an organization that delivers application services over the Internet, eliminating the need to have the application locally installed on each user's workstation.
B2B, B2C	Business-to-business and Business-to-customer; used in the context of targeting Web marketing efforts.
Bluetooth	Bluetooth is an emerging effort, spearheaded by the Swedish telecommunications giant Ericsson and sponsored by a large number of companies in consumer and business electronics and related industries, that is attempting to develop a very inexpensive universal Wide-Area Personal Network (WPAN) standard for appliances of all types. Bluetooth-enabled devices are expected on the market this year and are predicted to number in the billions within the next three to five years. Four IEEE standards efforts are afoot for Bluetooth: 802.15.1: 1MBps WPAN/Bluetooth 1.x derivative network 802.15.2: Recommended Practices For Coexistence In Unlicensed Bands 802.15.3: 20+ MBps High rate WPAN for Multimedia and Digital

Term	Meaning as used in this document
	Imaging 802.15.4: 200KBps WPAN for interactive toys, sensor and automation needs
CAD	Computer-aided design.
CGI	Common Gateway Protocol, a de facto standard that is universally implemented by HTTP servers.
CMS	Content management system. Can also mean Configuration Management System (see VCS for details).
CRM	Customer Relationship Management.
DAML	DARPA Agent Markup Language, an XML-based vocabulary that allows agents to share descriptions of network computing resources.
DARPA	Defense Advanced Research Projects Agency, sponsor of numerous technological initiatives including the Internet and DAML.
Datagram	A message carrying information from one machine to another as part of a scripted "conversation."
DBMS	Database management system.
FCC	Federal Communications Commission.
GNU	Free software initiative started in the 1980's by Richard Stallman; both a licensing model (as in GPL, the "GNU Public License") and a suite of Open Source software tools, most recently (and perhaps most significantly) the Linux operating system.
GUI, UI	Graphical user interface and user interface respectively.
HTML	Hyper Text Markup Language, a W3C specification.
HTTP	Hyper Text Transfer Protocol, an IETF specification.
IDE	Integrated Development Environment, a software tool that encapsulates compilers, interface design tools, and other aspects of application development under a seamless umbrella.
IEEE	Institute for Electrical and Electronics Engineers, Inc.
IEEE 802.11	Specification for IEEE relating to 1-2 Mbps Wireless LANs.
IETF	Internet Engineering Task Force, the standards body charged with the maintenance of the Internet.
IPX/SPX	Network protocol family associated with Novell Netware.
IT	Information Technology
Java, J2EE	Java is a programming language developed by Sun Microsystems, and Java 2 Enterprise Edition is an Internet-focused application framework

Term	Meaning as used in this document
	based on Java.
KBMS	Knowledge Base Management System.
LAN	Local-area network.
Linux	Open Source operating system developed in the 1990's that is a clone of the Unix operating system developed about 30 years ago by Bell Laboratories.
Mbps, Baud	Megabits per second and bits per second; measures of data connectivity bandwidth.
NetBIOS	Non-routable network protocol, very fast and simple, a flavor of which (called NetBEUI) is used widely in Microsoft network environments.
N-Tier	Multiple-tier application architecture in which two or more servers may be involved (e.g., a Web server and a DBMS server).
OASIS	Organization for the Advancement of Structured Information Standards.
OS	Operating system.
PDA	Personal Digital Assistant, examples of which include the Palm Pilot and Pocket PC product families.
RDBMS	Relational database management system.
RDF	Resource Description Framework, a W3C specification covering description of semantic networks.
T1, T3, OCS1, OCS3, ISDN, DSL	Standards for telecommunications connectivity (voice, data, or a combination) equivalent to multiple telephone lines. An ISDN (Integrated Services Digital Network) line is equivalent to two lines, a T1 is equivalent to about 28 lines, and a T3 is equivalent to about 780 lines. T1 and T3 assume copper wire; OCS1 and OCS3 are equivalent fiberoptic standards. DSL (Digital Subscriber Link) is a pooling mechanism allowing multiple users to share a single high-bandwidth connection. When used in the context of Internet connectivity, all of these except ISDN are considered high-bandwidth options.
TCP/IP, UDP/IP	Network protocols used ubiquitously on the Internet and more recently in corporate LAN/WAN environments.
VCS	Version control system; a file system for storing multiple versions of the same document, representing different states of the document at different times. Also sometimes referred to as a Configuration Management System, or CMS.
Vines	A network protocol developed by Banyan, used in a small but non-trivial minority of corporate computing environments.
W3C	World-Wide Web Consortium, a standards organization focused on

Term	Meaning as used in this document
	maintenance and enhancement of the World-Wide Web.
WAN	Wide-area network.
WAP	Wireless Application Protocol, and XML-based Internet transport protocol for Web-enabled devices such as cell phones. Analogous to HTTP for desktop and laptop devices. See also WML.
Wide-Area Personal Network (WPAN)	WAN designed for intermittently connected, relatively low-bandwidth, low power usage devices. See also "Bluetooth".
WML	Wireless Markup Language, an XML application that is the native protocol encapsulated in WAP messages. Analogous to HTML for desktop and laptop devices.
XML	Extensible Markup Language, a meta-language developed by the W3C for separating semantics from formatting in documents. The core of a large family of ancillary specifications and basis for a large number of domain-specific vocabularies.

References

Schwartz, Peter, *The Art of the Long View: Paths to Strategic Insight for Yourself and Your Company,* a Currency book published by Doubleday, New York NY, USA, 1991 and 1996. ***This book is the most accessible of all those listed here, and is a very enjoyable read!***

Ringland, Gill, *Scenario Planning: Managing for the Future,* John Wiley & Sons Ltd., Chichester, West Sussex, England, 1998.

van der Heijden, Kees, *Scenarios: The Art Of Strategic Conversation,* John Wiley & Sons Ltd., Chichester, West Sussex, England, 1996.

Selye, Hans, *The Stress of Life,* McGraw-Hill, New York, NY, USA, 1956.

Appendix: Introduction To Scenario Planning

Overview

Scenario planning is the art of storytelling applied to the future instead of the past or present. In this way it is not unlike science fiction—it's about "remembering the future." Unlike science fiction, though, the stories woven by a scenario planner revolve around a question or decision. This question or decision has been framed on behalf of a patron, usually an organization rather than a person. It reflects the focus of curiosity of the patron, who uses the scenario to improve strategic thinking by considering multiple possible futures.

SCENARIO PLANNING GIVES YOU MORE THAN A DRY STATISTICAL MEASURE OF A POSSIBLE FUTURE—IT GIVES YOU A PALPABLE SENSE OF WHAT IT WILL FEEL LIKE TO LIVE IN THAT FUTURE.

The benefit of scenario planning is not prescience, but flexibility and adaptability. Stewart Brand, creator of the Whole Earth Catalogs, has said that what scenario planning means is that while you may not always be right about the future, you are almost never wrong. And being wrong about the future is the only way to be unprepared.

Scenario planning gives you more than a dry statistical measure of a possible future—it gives you a palpable sense of what it will feel like to live in that future. Scenario planning requires a flair for the dramatic, a sense of how to help your audience suspend disbelief in a possible future that otherwise might never have occurred to them. But that possible future must still be demonstrably possible—it is not enough to be a convincing storyteller! To demonstrate this possibility, each future scenario must be carefully researched and documented. The better our research, the more convincing (and correct!) our scenarios will be.

The Process Of Scenario Planning

The process of scenario planning described here closely follows the method described in Peter Schwartz's book, *The Art of the Long View*. There are many ways of doing scenario planning, some others of which are described in the books listed in the References section (which also includes reference information on *The Art of the Long View*). We have to choose a place to start, and Schwartz's approach is well tested, well known, and easy to understand. In the author's opinion, it's also enjoyable. We highly recommend Mr. Schwartz's book if you find this paper useful!

We go through the following steps in the scenario planning process:

- Framing the question
- Researching the facts
- Identifying local forces
- Finding the driving forces

- Developing the matrix
- Choosing scenario plots

Scenario planning is not a spectator sport. Expect to get involved, even passionately involved, as the process unfolds.

Framing The Question

Finding the right question is a major step in the direction of the answer. Once you think you know the question, look for the questions behind the question, to make sure you have looked deep enough.

SCENARIO PLANNING IS NOT A SPECTATOR SPORT. EXPECT TO GET INVOLVED, EVEN PASSIONATELY INVOLVED, AS THE PROCESS UNFOLDS.

Examine your mindset. How do you habitually view decisions of this nature? Consider the possibilities - optimist, pessimist, blasé. Optimists believe every decision is a winner. Consider what pitfalls you are overlooking. Pay special attention to risk factors you assume to be minor. Is there a possibility they could be more real than you imagine them to be?

Pessimists, on the other hand, expect failure and are uncomfortable with visions of success. Are you overestimating risks or underestimating payoffs? Is the pessimism rooted in past failures? If so, is the present situation actually likely to suffer the same fate?

Surprisingly, though, the biggest potential pitfall is a simple belief in the status quo. Believers in the status quo approach (i.e., most of us) need to look carefully at their assumptions. Many times significant change has come about without much advance notice - or at least it seems that way to most of us, since we are adapted to the way things are and not expecting anything out of the ordinary.

This can be true even of situations that were until very recently unthinkable. Do you, for example, now take cellular phones and the World-Wide Web for granted? Most of us do, but cell phones did not exist 20 years ago and were relatively rare a decade ago. David Gelernter, a computer scientist at Yale, is considered a visionary because he saw the Web coming a mere five years in advance, in 1987! To shake up our belief in the inevitability of the status quo, we need to uncover our assumptions, which are often not very obvious to us.

We will be looking at this again when we get to the search for driving forces. For now we need to look at the question we are asking and consider carefully how the question might be different - or even irrelevant - if one of our underlying assumptions proved incorrect.

Keep the question focused on events over which you have no control. Otherwise you can change the outcome simply by changing your decision. Remember, you are looking to predict the future, not to decide it.

For example, you might be wondering whether to invest in biotechnology stocks. Suppose instead we try framing the question as "What would cause the price of biotechnology stocks to rise or fall significantly in the next three years?" This keeps the

question focused on events over which we have no control, and hence more need of understanding.

Other key techniques in framing the question:

- Look at broader and narrower questions to refine the focus.
- Try to identify key indicators that will tell you which way a question is going to go.
- Clarify your time frame when framing the question. The answer may be very different if you are looking at a one-year rather than a five-year window.
- Consider whether you need to look locally or globally in choosing your focus. Small businesses usually need only look at local issues, and at local expressions of global issues.

CLARIFY YOUR TIME FRAME WHEN FRAMING THE QUESTION. THE ANSWER MAY BE VERY DIFFERENT IF YOU ARE LOOKING AT A ONE-YEAR RATHER THAN A FIVE-YEAR WINDOW.

Having considered the dangers of optimism and pessimism, and examined our assumptions about the status quo, we are ready to look for driving forces.

Researching The Facts

Once the question is framed, it is time to find out everything you can about the subject and its surrounding context. The goal is twofold: learn what can be learned about the subject at hand, and educate yourself to improve your ability to ask the right questions.

Once again it behooves us to consider our perceptual biases. We need to be aware of the outcomes we expect and look for evidence to the contrary. This is based on a fundamental tenet of the scientific method - the attempt to disprove the null hypothesis.

Consciously adopt different mindsets. If you are a businessperson try to see the world as a politician would see it. If you are over 30, make a determined effort to see the world through the eyes of a teenager.

We continually have to filter our perceptions because of the information overload we experience. The object here is to make sure we stay aware of our biases and make sure our fact-gathering processes let in enough variety of information without becoming overwhelmed.

What areas of inquiry do we pursue when doing our research? The question determines this in large measure, but there are some general considerations that should be taken into account. Every researcher has his or her own favorite "usual suspects." Mine include the Economist (http://www.economist.com) and caffeinated-beverage-consumption sessions employing the Google search engine (http://www.google.com), which is also accessible from your Web-enabled handheld (http://www.google.com/palm/). I often find useful information in Usenet newsgroup archives as well.

The Greenspan Factor

One icon of U.S. society who cannot be ignored is Federal Reserve Board Chairman Alan Greenspan. Greenspan is, by the nature of his job, a futurist. Perhaps we can learn something useful to our scenario planning from his techniques for gathering facts. *These are presented here only as an example, and given his particular career path his areas of interest may be very different from yours!*

A recent magazine article (*b*[3], 11-12/2000, p. 16) listed seven areas Greenspan continually monitors in his fact-gathering processes. Focusing on these may shift our perspective in fresh and surprising ways:

Process of innovation	What areas of innovation are especially fertile relative to the question at hand?
Creative destruction	This concept, which was coined by the economist Joseph Schumpeter, refers to the continuous process of emerging technologies making existing technologies obsolete and pushing them out of the way. Are there any such processes at work with respect to the question under consideration?
Expansion of knowledge	What kinds of new knowledge could affect the question, pushing it in one direction or another over the time period under consideration?
Productivity gains	To what degree will an increase or decrease in productivity of one kind or another affect the outcome? What would need to be measured to assess the direction and magnitude of productivity gain or loss?
Continued US dominance	To what extent does the outcome depend on US dominance of a particular market or technology?
Job insecurity	What effect will the increasing relative insecurity of the average US worker's job have on the question at hand? Does the outcome depend on this insecurity continuing, increasing, or abating?
Education	What affect will the education system have on the question at hand? Does the outcome depend on changes in the system, such as a trend toward Internet delivery of education? Does the outcome assume a certain level of literacy/numeracy in the population?

BY STARTING LOCALLY, WORKING OUTWARD FROM THE QUESTION ITSELF, WE ENSURE THAT THE FACTORS WE IDENTIFY ARE TRULY RELEVANT. WE WORK OUTWARD LIKE RIPPLES CAUSED BY THE DROP OF A PEBBLE IN A POND.

Identifying Local Forces

Having decided on the question and done our research, it is time to identify the local forces that will affect our decision. The secondary question we are trying to answer here is, simply put: What will tell us that the main question has been answered?

By starting locally, working outward from the question itself, we ensure that the factors we identify

are truly relevant. We work outward like ripples caused by the drop of a pebble in a pond. For each factor we locate, we ask ourselves, "What then are the forces driving that factor?"

Eventually we reach a point where the factors are no longer so local. The exact point is not always clear (nor does it need to be), but eventually we shift into the next stage, the search for driving forces.

Finding The Driving Forces

Next we look for the global factors that underlie the local forces. These factors typically are drawn from five main categories of sources:

- Society
- Technology
- Politics
- Economics
- Environment

Clearly there are interrelationships, but this list provides a viable starting point for the search. The nature of the question may suggest additional areas, or refinements of the above-listed areas.

What are the driving forces? They are the trends and currents at work in the larger world that will determine the direction the local forces take. In short, driving forces are the larger issues that determine the framework within which the local forces operate.

For example, if the question at hand is whether to start a new business, the state of the overall economy is a driving force. If your business will involve marketing to C++ language programmers, the IT industry shift to server-based Java environments is a driving force - as is the burgeoning Palm Pilot user base, since C++ is the language of choice on the Palm OS platform.

The identification of driving forces will proceed smoothly if you have done your homework with diligence. Keep in mind that you are looking for forces beyond your control. Just as the question itself should not depend on a decision on your part, the driving forces should be larger than your scope of activity can encompass.

Some driving forces are predetermined by nature, and others face futures that are anything but certain. The number of United States citizens who will be eligible to vote in the 2004 Presidential election is easy to predict, since the vast majority of such persons are already citizens, and the number of persons who can be naturalized by that time is fixed, since they must already have applied if they are to complete the process in the next four years. The variable factors include the death rate and the number of citizens who choose to renounce their citizenship, both of which are at least loosely predictable.

The number of those eligible who will actually vote, on the other hand, is uncertain, since it depends on how many of those who are eligible actually register, and how many of those who are registered actually turn out to vote.

If the question at hand depends on the number of actual voters rather than the number of those who are eligible, the registration rate and the turnout rate represent critical uncertainties. Both numbers (those eligible and those who actually vote) are driving forces, but they will have very different effects on the scenarios we develop.

UNCERTAINTY IS NOT NECESSARILY CRITICAL TO THE OUTCOME. IF A FACTOR CAN COME OUT IN TWO VERY DIFFERENT WAYS AND THE QUESTION STILL WORKS OUT TO THE SAME RESULT, THE UNCERTAINTY IS NOT CRITICAL.

Note that uncertainty is not necessarily critical to the outcome. If a factor can come out in two very different ways and the question still works out to the same result, the uncertainty is not critical - discard it as irrelevant.

A situation may seem to have no critical uncertainties, as if everything about the situation were predetermined. In this case the odds are excellent that you are either overlooking some important factor, or that your predetermined elements are more mutable than you think. Ask yourself under what conditions your assumption of constancy might prove incorrect. Be open to the possibility that your assumptions could be wrong. This is a great source of possible candidates for critical uncertainties.

Coming out of this step, we have two lists: the predetermined elements and the critical uncertainties. In the next step, we will use these to construct a matrix that will lay bare the framework in which our scenarios will take root.

Developing The Matrix

The matrix is not an absolute necessity in scenario building, but for us newcomers it is a good place to start. The goal of this step is really the selection of the plot lines our scenarios will follow, which is what the next step is for; if the plot lines are already obvious to you feel free to jump ahead.

The matrix is like a Cartesian coordinate system, with the two most salient driving forces as the X and Y-axes. The four corners of the coordinate system represent four possible outcomes; normally only two or three will actually turn out to be viable combinations, forming the basis for our scenarios. The predetermined elements are the backdrop to the matrix, the constant features of the terrain as it were.

A hypothetical example may help clarify. Suppose we have determined that the two most critical uncertainties for us are the state of the world economy and the weather in Central America. Our matrix uses these as its axes. For the sake of simplicity, we are using 'good' and 'bad' as our names for the ends of both axes.

Now we have four combinations:

- Good weather/good economy
- Good weather/bad economy
- Bad weather/good economy
- Bad weather/bad economy

We can construct a scenario for each of the combinations, though it will be to our advantage to limit our final result to two or three. Most likely it will turn out that one of the combinations will turn out to be either very unlikely or not very interesting.

Choosing the scenarios is the next step in our progression.

Choosing Scenario Plots

Once we have identified the driving forces and determined how they interrelate, we start the search for plots that convincingly portray possible futures that reflect the outcome of particular combinations of driving forces.

Every good plot has characters; in scenario planning we must keep in mind that the characters are not persons. Instead the characters will be "larger than life"—institutions like corporations, governmental bodies, or even entire industries; ecological forces like global or regional weather; mass entities like the population of eligible U.S. voters or high school age males; or societal trends like nationalism or religious fundamentalism. There may be a single individual who seems to be a major player, such as a national leader or religious figure, but such an individual is almost always a creation of some larger trend in society. If the individual is struck down the trend continues, and another individual emerges to fill the role. Hence it makes the most sense to cast the trend as the character rather than the individual.

THE ORGANIZATIONAL CHARACTERS IN A SCENARIO PLOT ARE MOTIVATED BY MANY OF THE SAME "EMOTIONS" THAT WOULD MOTIVATE A HUMAN CHARACTER.

The organizational characters in a scenario plot are motivated by many of the same "emotions" that would motivate a human character. A plot develops when characters conflict with one another and synergize with still others. As Hans Selye demonstrated in his seminal work *The Stress of Life,* stress induces the reactions that cause adaptation to occur in an organism; it's no different in the lives of institutions like corporations and governments. In choosing a scenario plot, we need to ask ourselves: What stresses do the driving forces represent and how will we react to them?

The organizational characters in a scenario plot are motivated by many of the same "emotions" that would motivate a human character. A plot develops when characters conflict with one another and synergize with still others.

In The Art of the Long View, Peter Schwartz described several standard plot types that occur over and over again, and a few others that pop up from time to time. They are well worth your consideration when engaged in plot divination.

Common plots include the following:

Winners and losers	The world is a zero-sum game. The players are enemies in a dog-eat-dog world.
Challenge and response	Like the Seven Labors of Hercules, the world consists of monumental obstacles to overcome. The protagonist (presumably the session sponsor) is a hero who must draw on deep wells of

	creativity and courage to prevail.
Evolution	The world unfolds in a relatively consistent upward spiral. Technology-oriented scenarios will often follow this plot, due to the commonplace path taken by scientific and engineering thought.

Less common, but perhaps more interesting, are these:

Revolution	Discontinuity is the hallmark of a revolution-based plot. The emergence of the World-Wide Web is a plot that diverged from the usual evolutionary path of technology. Few people foresaw the Web, and the few who did had no advance idea of the scale or rapidity of its success.
The Lone Ranger	The protagonist stands alone against the world. Key elements of the Lone Ranger plot include the protagonist standing up for what is Right, and the Villain (usually the competition) who is looking out for its own interests rather than the common good.
Cycle	Some plots, especially those based on economic or demographic trends, are inherently cyclical in nature.

The bottom line: you need to find a plot that more or less matches the situation described by a particular combination of critical uncertainty outcomes. The next step is to flesh out each scenario with a narrative that draws the listener in and convincingly presents the scenario logic.

Fleshing Out The Scenarios

Creating a compelling narrative is the most creative aspect of this process, and therefore the aspect least susceptible to the high-minded dispensation of advice. However, there are some points you can keep in mind that come from the world of creative writing:

- Character development is important to readers. While the characters in the plots are larger than life, you can often find a way to bring the plot down to a human level by showing the effects of the scenario on a day in the life of a hypothetical person or persons. Who lives in this world? What can you say about their lives to make your prospective audience care about them?
- Action is good. Stories should contain something more than "talking heads." In any good story there are usually antagonists as well as protagonists, and antagonism is not about good or bad—it's about the conflicts that inevitably arise in people's lives, or in this case the lives of organizations, etc.

  However, you can describe how these conflicts would be manifested in the lives of people. What happens to the protagonists/antagonists as a result of the operating of the driving forces? How do the forces change them and the world they live in?

Presenting Your Scenarios

Once the scenarios are complete, you are ready to present your scenarios to the sponsors. If you have done your work well, the scenarios will be controversial, but also believable.

If they are dismissed out of hand, a number of possible explanations exist:

- The scenarios are too scary
- The scenarios are too far-fetched
- The audience isn't ready for new ways of thinking

PRESUMABLY SOMEONE IN TOP MANAGEMENT IS INTERESTED IN PROMULGATING NEW WAYS OF THINKING, OR YOU WOULDN'T BE DOING SCENARIO PLANNING IN THE FIRST PLACE!

You can solve the first two problems by dint of your own efforts, but the third requires intervention and support by top management. Presumably someone at that level is interested in promulgating new ways of thinking, or you wouldn't be doing scenario planning in the first place! They need to propagate that interest to the rest of their staff. You can help by providing ammunition in the form of explanations, success stories, and so on; but the ultimate responsibility remains with the sponsor for convincing the staff of the need for more open ways of thinking.