

Application Server Platforms

Converging Middleware Solutions for Integration and Development

Abstract

Application Server Platforms: Converging Middleware Solutions for Integration and Development

The growth of e-business is driving the demand for middleware solutions that seamlessly integrate disparate systems into a unified value chain. A profusion of middleware categories complicates understanding and acquisition of solutions. A review of selected literature from January 1997 to June 1999 of the larger area of distributed computing identifies trends in the marketplace toward consolidation of solutions into value-added platforms.

Conclusions are presented as recommendations for IT professionals and managers to guide decisions about middleware.

Table of Contents

Abstract *

Table of Contents *

List of Figures *

Chapter 1. Introduction *

Brief Purpose *

What is Middleware? *

Full Purpose *

Significance of the Study *

Limitations to the Research *

Problem Area of the Paper *

Chapter 2. Review of References *

Business Publications *

IT Research Firm Advisory Publications *

IT Industry Publications *

Chapter 3. Methods *

Data Collection *

Data Analysis *

Data Presentation *

Chapter 4. Analysis of Data *

The Middleware Market *

Distributed Component Architecture *

One-Tier *

Two-Tier *

Three-Tier *

Benefits of Distributed Components *

Communication Methods *

CORBA *

COM *

Enterprise JavaBeans (EJB) *

Message-Oriented Middleware *

Middleware Infrastructure *

Application Servers *

Trends in Distributed Component Technology *

Application Integration Solutions *

Web Integration *

Chapter 5. Conclusions *

Recommendations *

Appendix A *

Definitions *

References Cited *

List of Figures

Figure 1: Distributed Component Architecture [_*](#)

Figure 2: CORBA Architecture [_*](#)

Figure 3: COM/DCOM Architecture [_*](#)

Figure 4: Enterprise JavaBean Architecture [_*](#)

Chapter 1. Introduction

Brief Purpose

Today, electronic commerce is driving the demand for middleware solutions to enable the seamless integration of distributed systems into a unified value chain (Gilpin, 1998) (Brown, 1999). Cross-enterprise integration is a business problem. It is a new problem brought on by the unprecedented scale, scope, and complexity of the applications being attempted today (Papows, 1998). Middleware is essential to information technology (IT) initiatives like e-commerce because it provides the services and protocols necessary to tie disparate systems and architectures together.

The purpose of this report is to help IT professionals understand the different categories of middleware so they can select the best solution for their needs. In addition, this report is designed to put middleware in context for IT managers as they define their technical architectures. The findings of this study are presented as a list of recommendations that are meant to simplify choices for IT professionals and to provide IT managers a perspective to help ease the decision making process.

What is Middleware?

Middleware is such an integral component of distributed computing that it is difficult to describe it in terms of a single concrete definition. (Schulte, 1997) refers to middleware as the "glue" that holds all components of distributed applications together. (Magrassi, 1997) refers to it as "an enabling layer of software that resides between the business applications and the networked layer of heterogeneous platforms and protocols" (p.1). Vendors refer to middleware by various name transaction servers (Microsoft), application servers (BEA Systems), component servers (Sun), and business rules servers (IBM). Because of the variety of terms and definitions used to talk about middleware, a "stand-alone" listing is located for quick reference in this paper (See Appendix A).

In this paper, the term "middleware" is used to refer to a set of system services and protocols that map applications to the resources they use in a distributed multi-tier client/server environment. However, regardless of the name or the terminology, the same set of services is required to deliver the full functional capabilities of a distributed component architecture. These capabilities include scalability, adaptability, reliability and manageability (Berson, 1996), (Orfali, Harkey, and Edwards, 1997), (Edwards, 1999).

Full Purpose

Today, electronic commerce is driving demand for middleware solutions to enable the seamless integration of disparate systems as enterprises are forming Internet-based trading groups, integrated supply chains, and virtual companies to create a unified value chain (Gilpin, 1998) (Brown, 1999). However, while the current technologies allow developers to create applications and business processes that span multiple systems, the integration process is still extremely complex.

In a recent survey of Fortune 1000 companies, Forrester Research Inc. found that 73% had to use a combination of middleware technologies along with custom programming to deploy real-time application integration solutions (Brown, 1999). As the focus of integration shifts to Internet applications, users will demand solutions that manage enterprise-scale processes more effectively (Sweat, 1999). In light to these demands, middleware takes on increasing importance as well as a new role in the development, deployment, and management of applications and resources.

In order to help IT professionals understand the differences between categories of middleware that can support e-commerce, the trends driving

the emerging market for distributed component technology, application servers and application integration solutions, the purpose of this paper is three-fold:

- To define a middleware architecture that will support the high volume distributed transaction usage pattern required for real-time application integration.**
- To examine the middleware categories that have the functionality to support that type of usage pattern, including Component Object Model (COM), Common Object Request Broker architecture (CORBA), Enterprise JavaBeans (EJB), application servers, transaction processing (TP) monitors and message-oriented middleware.**
- To examine the requirements for integrating Web-based applications into the enterprise.**

The results of each of these three investigations are presented in the form of a list of recommendations that can be used by IT professionals so they can select the best-integrated solution for their needs, and provide IT managers with a framework for making strategic investment decisions in middleware technologies. The goal of this paper is examine the issues surrounding integrating enterprise applications and how integration fits into a company's technical architecture considerations via a unified strategy for application integration and development.

A literature review focusing on identification of trends in usage of middleware solutions to enable IT initiatives like electronic commerce was conducted in order to describe the larger context and need for this study. Literature was collected from January 1997 to June 1999. That time period was chosen because it parallels the rise in distributed computing with N-tier client/server architecture — an applications development approach that partitions logic across three or more environments: the desktop, one or more application servers, and a database -- as the dominant form (BEA, 1999) (See Appendix A). The selected published materials include:

- Strategic analysis reports, enterprise architecture planning guides, and distributed platform research papers from Gartner Group, Giga Information Group, and Forrester Research, three leading IT research firms. These technology research firms were selected because they specialize in advisory services on a wide range of next-generation**

technologies and architectures.

- **Case studies of four companies using these technologies to support their core business processes. These cases were selected because they involve production systems that support high volume secure distributed transactions.**
- **Technical articles and conference position papers on middleware products and strategies that support a high transaction and high security usage pattern.**

In addition, an emergent content analysis approach, based on grounded theory research design (Strauss and Corbin, 1990), was utilized to analyze the literature in order to identify selected information relative to application server platforms for distributed component applications and application integration solutions. Additional information relating to middleware technologies from four vendors: Microsoft, IBM, BEA Systems, and Sun was also reviewed for this purpose.

Significance of the Study

Application integration solutions enable an infrastructure-based approach that reduces the cost, complexity, and interoperability issues associated with merging disparate systems (Berson, 1996) (Weil and Broadbent, 1998). Without an integration infrastructure many interfaces have to be built to establish communications between the various components of the distributed system, which are expensive to create and maintain. Establishing a more uniform approach to moving information and events between applications reduces the cost and complexity of application integration and improves responsiveness of the collective set of applications to the needs of the business (Orfali, 1997) (Edwards, 1999).

There is a complex array of choices in middleware and integration technologies on the market today. The paper is designed to be helpful to IT managers and professionals who are either responsible for infrastructure investment decisions or trying to differentiate and properly position a variety of middleware solutions that can support IT initiatives like e-commerce. The findings of this paper are meant to simplify those choices.

Limitations to the Research

The literature review was limited to January 1997 to June 1999. The concept

of middleware grew out of the challenge to make client/server technology work effectively to solve business problems on an enterprise scale. As companies discovered that two-tier client/server systems don't scale well enough to support a large number of users, high transaction volumes, and the unpredictable workloads of the Internet, they moved their applications to a three-tier model.

Three-tier architecture distributes the processing load between (1) the clients that run the graphical user interface, (2) a middle tier of protocols and services (middleware) running the business logic, and (3) the database and/or legacy application. A three-tier architecture deploys application components (presentation, functional logic, and data) across three tiers of computer platforms: desktop machines, intermediate application servers, and back-end database servers.

This architecture became the dominant client/server model in 1997. Therefore, the research timeframe was selected to coincide with the dominance of this model and chronicles the changing role of middleware technologies to date with a focus on distributed components and application integration solutions. (Berson, 1996), (Orfali, Harkey and Edwards, 1997), (Edwards, 1999)

Information relating to middleware technologies from four vendors: Microsoft, IBM, BEA Systems, and Sun was accumulated from publications between January 1997 and June 1999. These vendors were selected because their technologies have been widely documented by industry analysts in multiple publications. They have also been identified as the vendors today that appear to be in the best position to emerge as the leaders in the distributed component market.

Consolidation of the middleware market has combined several categories of middleware into higher value solutions. Many categories and products available six months to a year ago are no longer on the market today. Therefore, this paper will not attempt to identify vendor products but rather define the elements that should be considered when selecting middleware technologies.

Problem Area of the Paper

Middleware is essential to information technology initiatives like electronic commerce and interactive Web-based applications on the Internet because it

provides the services and protocols necessary to tie disparate systems and architectures together. In this era of intensely competitive global commerce, every business is becoming a technology business. And the success of any organization is increasingly determined by its ability to unlock the potential of a variety of rapidly emerging technologies, particularly those for the Internet. In a recent *Business Week* article (Byrnes and Judge, 1999) said,

"There's a revolution under way, and mastering the Net has moved front and center on Corporate America's agenda. The Internet model, with fewer hard assets, a direct pipeline to customers, and freedom from the hierarchical management structure of most of Corporate America, offers a new level of speed and operational efficiency for those who master it — and huge dislocations for those who don't" (p. 80).

The Internet is transforming the economy and the way business is conducted as Net-based companies are restructuring the corporate landscape (Byrnes and Judge, 1999). In an *Information Strategy* (1998) global Internet survey special report Novell chairman, CEO, and survey sponsor Dr. Eric Schmidt said,

"Use of the public Internet for private business is a revolutionary change producing huge cost savings and competitive advantage for the businesses quick to adopt it. The Internet is driving communication and collaboration both within and between companies, advancing globalization providing cost-effective channels for electronic commerce. By 2002, according to the best projections, worldwide Internet commerce will likely be a \$300 billion business" (p. 2).

(Byrnes and Judge, 1999) show that Schmidt's 1998 projections were slightly conservative in their *Business Week* article when they wrote,

"Almost overnight, the Net has become a huge part of the economy. A recent study by the Center for Research on Electronic Commerce at the University of Texas, commissioned by Cisco Systems, Inc. found that the Internet economy grew at an astonishing annual 174.5% rate from 1995 through 1998. At \$300 billion today, that rivals sectors like telecommunications and autos" (p. 81).

In light of these facts, for many companies moving operations "online" is a matter of survival. (Gilpin, 1998), (Dec, Mack, and Anderson, 1998), (Papows, 1998) and (Weil and Broadbent, 1998) identify similar emerging patterns. As electronic commerce grows, a major change in the way business is being conducted is taking place. Many enterprises are rapidly moving to global commerce, electronically linking customers and suppliers around the world.

In effecting this change, Web-based applications are being integrated across multiple companies. Today, almost every part of the business and its associated applications has been affected by the Internet, or is about to be. (Papows, 1998) identifies the following types of cross-enterprise integrated applications use as examples (p.83):

- *Electronic market solutions* enable communities of suppliers to organize around customers and compete for their business. (Hof, McWilliams, and Saveri, 1998) and (Kolesar, Van Rysin and Culter, 1998) provide detailed examples of these types of business-to-business solutions in their articles.
- *Alliance operation solutions* enable communities of organizations to collaborate for competitive advantage. (Freidheim, 1999) examines the evolution of these types of solutions in his article "The Trillion Dollar Enterprise".
- *Communities of interest and interactive distributed learning solutions* enable groups of people with common interests to collaborate and participate in joint discovery, collaborative training, and decision-making without regard to their geographic location. Refer to (Pasternack and Viscio, 1998) "The Centerless Corporation" for more perspective.

To support these solutions, the architecture behind the Web site has to undergo an essential transformation too. Architecture is now rapidly becoming "mission-critical infrastructure" because it dictates the way in which an application is created and how its components are distributed across multiple systems (Gilpin, 1998). As the IT industry transitions from centralized mainframes to distributing computing, the challenge is to integrate new systems with legacy systems to meet the needs of the business (Dec, Mack, and Anderson, 1998). The economic imperatives of the cross-enterprise are now driving an acceleration of this integration (Gilpin, 1999)

Chapter 2. Review of References

The review of reference section is organized like an annotated bibliography -- each entry contains a brief description summarizing or describing content. The references in this section are organized under three

publications category headers: (1) Business Publications, (2) IT Research Firm Advisory Publications, and (3) IT Industry Publications.

A brief description appears under each category header describing the scope of the sources listed and the significant topics included. Key aspects of the entries relative to the purpose and problem of this study are identified.

Business Publications

This group of references identifies the economic imperatives for real-time application integration solutions. The publications under this header discuss the impact of the Internet on corporate America and the collection is designed to acquaint managers with the opportunities and risks associated with real time application integration initiatives like e-commerce. Topics: Strategy, management and competition, Internet applications, global enterprise collaborations, market-facing enterprises, knowledge management, and customer relationship management. These references define the business problems driving the demand for new middleware solutions.

Byrnes, Nanette, and Judge, Paul, (1999, June 28), "Internet anxiety", *Business Week*, page 78-88, McGraw-Hill.

Byrnes and Judge discuss the trend in corporate America to embrace a new business model based on the Internet. It examines what traditional companies in various industries (entertainment, transportation, banking and financial service, etc.) are doing to move their operations "online" as a matter of survival. Provides a good synopsis of how companies are using the Internet to gain a competitive advantage.

Freidheim, Cyrus F., (1999, First Quarter), "The trillion dollar enterprise", 30 pages, *Strategy and Business*, Booz Allen and Hamilton, Reprint No. 99106.

Freidheim's proposition is that huge global networks and alliances will soon create organizations of awesome capabilities. He proposes that,

"... They will not be the conglomerates of the 1970's and 1980's that attempted to gain stability by diversifying risk, spreading their investments over separate businesses. Rather, these new enterprises will focus all their resources on dominating one, or a few, fields. They will restructure whole industries, change their economics and turn the basis of competition upside down. They will, collectively, have all the capabilities necessary to

win in a tough, competitive, global marketplace" (p.1).

Some of the major benefits organizations can expect to realize from such alliances are (1) Increased flexibility and speed, both from a variable cost standpoint and in delivering business solutions. (2) Access to new technologies and skills. (3) Work elimination through differentiated and/or reduced service levels. (4) Ongoing productivity improvement, operational reliability and infrastructure, and (5) enhanced training and development opportunities for IT employees.

Hof, Robert, McWilliams, Gray and Saveri, Gabreille, (1998, June 22), "The click here economy", *Business Week*, page 20-29, McGraw-Hill.

The Internet is stimulating business transactions by making them easier and cheaper to conduct. The authors state, "Internet commerce is not a product but a force, a powerful agent that will transform the way nearly every product and service is created and sold" (p.2). This article examines both business-to-business and business-to-consumer transactions on the Internet and concludes that market opportunity for manufacturers, re-sellers, service providers and vendors is enormous, but so are the challenges in this ever-changing market. Challenges are many as intensification of the competitive environment spurs consumers to expect better service quality and pricing.

For many consumers price isn't all that matters. For a nation of shoppers, there is real value in being able to shop at any hour, for almost anything, anywhere in the world. The long-term gains from doing business on the Internet are incomprehensible.

Papows, P., (1998), *Enterprise.com. Market leadership in the information age*, Perseus Books.

In this book Jeffrey P. Papows, Ph.D., president and chief executive officer of the Lotus Development Corporation examines the state of the Internet and e-commerce. He discusses the potential of information technology to change whole industries through the use of the Internet. Additionally, his discusses the concept of the "market-facing enterprise" -- an organization in which relationships and functions are enhanced through the use of technology. Other topics include knowledge management and how the adoption of new technologies is changing the roles and responsibilities of educators, researchers, and policy makers.

Pasternack, Bruce, and Viscio, Abert, (1998, Third Quarter), "The centerless corporation: A model for tomorrow", page 15-35. *Strategy and Business*, Booz Allen and Hamilton.

High-performance companies are structured around core values, knowledge, and people. They know what they stand for and how to share services. That is the proposition of this article, which uses a hypothetical corporation as a model for competitive advantage. The authors invite the reader to imagine a corporation that is built around resources, (people, knowledge, and capabilities) rather than the assets that get lined up on financial balance sheets. "A Corporation that manages its people with a relationship we called "new people partnership" rather than downsizing lists and that places great stock in its knowledge department and its chief knowledge officer. Its interdependence rather than the independence of its parts characterize this corporation. This is the Centerless Corporation" (p.6). Also, examines how information technology creates real value for the corporation.

Special Report, (1998, November), "The global Internet 100 survey 1998", *Information Strategy*, [Online], 20 pages. Available: <http://www.info-strategy.com/G1100/index.html> [November 15, 1998]

This report publishes the results of a comprehensive survey undertaken by several university media and information technology directors to determine how much business the world's biggest companies are really conducting over the Internet. The results show that nine out of ten top ranking firms conducting business on the Web are in the USA. The one exception was the number two ranked Lufthansa Group in Germany. Excellent overview of how the Internet is being used globally.

Weil, Peter, and Broadbent, Marianne, (1998), *Leveraging the new infrastructure: How market leaders capitalize on information technology*, Harvard Business School Press.

To remain competitive, organizations must harness and leverage the benefits of emerging networking and computing technology. They must develop business strategies that are understood and supported by executive management. In this book Weil and Broadbent present a thorough and practical approach that connects business strategy and the value of information technology investments.

IT Research Firm Advisory Publications

This category of references focuses on publications from three IT research firms, the Gartner Group, Inc., Forrester Inc., and the Giga Information Group. This selection of publications provide the foundation for analysis of the middleware market, middleware architecture, and identifies emerging trends in distributed component technology and application integration. Topics: collaborative computing, middleware architecture, distributed platforms, cross-enterprise application integration, middleware, and Web-based application integration.

Anderson, M., (1998, 25 March). *The IEW marketplace: Key trends and electronic workplace architecture*, [Online], # pages. Available: GartnerGroup Interactive. <http://www.gartner.com> [May 28, 1999]

Anderson discusses five key trends in the electronic workplace architecture that are influencing technologies, applications development, and enterprise integration solutions. Main focus is on standards and architecture requirements for a distributed computing environment.

Brown, Eric G., (1999, March), *Integrating Business Processes*, Forrester [Online], 27 pages. Available: Forrester. <http://www.forrester.com> [May 25, 1999]

Eric Brown and his staff interviewed 30 information technology executives from Fortune 1000 firms for this Forrester report to find out -- the types of integration projects they are deploying; which technologies they use to integrate their applications, and what are the greatest challenges they face today in doing real-time integration. The primary findings of those interviews are:

The primary business drivers for their integration projects were e-commerce, customer relationship management and enhanced supply chain operations. Clients are building on-line virtual stores; deploying call center for customer support, and optimizing supply chain management systems. Each of these systems requires real-time application integration across multiple disparate systems. The integration technologies used are -- 73% are using messaging, 43% are using direct connections, and 3% are using message brokers. However, none of the solutions meet the process integration -- implementing and

managing transactions and real-time business processes that span multiple application -- requirements. Currently they must build customized solutions, which is complex and costly.

Costa, Philip, (1999, January 6), *The web and distributed components*, [Online], 9 pages. Available: Giga Information Group, <http://www.gigaweb.com> [May 15, 1999]

This article provides a good overview of the market evolution to technologies to support component-based development. The tools available to support Web-based development are currently divided into five categories. Each category is rated according to three criteria legacy integration, middle-tier development (load balancing, connection pooling, failover and runtime services), and client development, the ability to develop a rich graphical user interface.

Costa, Philip, (1999, February 16), *Outlining the benefits of a layered application architecture*, IdeaBytes, [Online] 2 pages. Available: Giga Information Group <http://www.gigaweb.com> [June 1, 1999]

Costa examines the benefits of layered application architecture. A number of forces are driving the adoption of this architecture, but the two most notable are the integration of Web applications with existing enterprise applications; and the increasing need to integrate data managed by packaged solutions and developed application packages into a single logical view. Also, Costa identifies the five reasons users are adopting layered application architecture: flexibility, integration, scalability, maintenance, and security.

Dec, K., Mack, E., and Anderson, E., (1998, July 24), *From mainframes to distributed computing: The technical issues*, Strategic Analysis Report, Gartner Group, Inc., Spectrum for IT-Americas, document # R-05-4102.

This 100 page report discusses the technical issues of the IT industry's transition from mainframes to distributed computing. It describes the current IT environment as complex and fluid and discusses in detail the complexity involved in today's distributed environment due to the expense, and scalability issues surrounding application integration solutions because of the immaturity of the available technologies.

Enslow, B., and Schulte, R., (1998, August 19), *Outflank the competition by developing IT to build a 'zero-latency enterprise'*, Inside Gartner Group,

[Online], 9 pages. Available: GartnerGroup Interactive
<http://www.gartner.com> [June 2, 1999]

Enslow and Schulte purpose the concept of "zero-latency". In essence it states that enterprises that act quickly on new information have a competitive advantage, and "zero-latency" information strategies aim to push that advantage to its limits. The article describes five components of that strategy, which include detailed analysis of the implementation of real-time cross-enterprise application integration and process flow. It also explores how companies are utilizing these systems to improve customer value, speed cycle times, and create a more flexible computing environment.

Gilpin, Mike, (1998, December 30), *Distributed application platforms (thematic planning assumption*, Enterprise Planning Assumptions Guide, Giga Information Group, document # T-1098-002.

This planning guide describes the need for standardization of communications to ensure the greatest interoperability between application platforms. The primary focus of this report is on distributed application architecture. Topics include the trend toward thin-client computing, Java platform choices, integrated development tools, and industry consolidation in the middleware market.

Gilpin, Mike, (1999, March 30), *Giga proposes ideal middleware architecture: Future direction for industry and customers*, Enterprise Planning Assumptions Guide, Giga Information Group, document # P-1298-009.

Gilpin defines the requirements for ideal middleware architecture, according to its distribution over the layers of a typical distributed-application architecture. He presents this ideal as an abstraction that could support a physical topology of any number of tiers. He defines the requirements for a client tier, middle-tier business logic and process, and a services-tier, also known as the data access tier.

Gilpin, Mike, (1999, May 10), *Internet application integration: A new market to support cross-enterprise e-business*, Enterprise Planning Assumptions Guide, Giga Information Group, document # P-0599-005.

Trends in usage of enterprise application integration solutions enable e-business and signal an evolution of the integration market into a new phase, Internet application integration. In this article Gilpin discusses

what is required from integration solutions when integration takes place across multiple enterprises over the Internet.

Gilpin, Mike, (1999, May 27), *Application servers are primary area of future middleware research interest for GigaWorld Europe attendees*, IdeaBytes, [Online] 3 pages, Available: Giga Information Group <http://www.gigaweb.com> [June 1, 1999]

Gilpin discusses the value proposition of an application server compared to other alternatives and examines the trends for future usage. The core value proposition of an application server is to enable much easier and more rapid development, deployment, and management of Web and other types of distributed applications.

Magrassi, P., (1997, April 2), *Middleware: Dispelling the fog*, Technology Research Notes, [Online], 8 pages. Available: GartnerGroup Interactive. <http://www.gartner.com> [May 10, 1999]

This article provides a good introductory overview of middleware. It describes the main categories of middleware and the primary uses for each category type

Schulte, R., (1997, December), *Middleware: The glue that holds distributed computing together*, Enterprise Architecture Planning Group, [Online], 10 pages. Available: GartnerGroup Interactive <http://www.gartner.com> [May 30, 1999]

Schulte describes the various categories of middleware and explains why middleware is the essential foundation for distributed computing.

Schulte, R., and Natis, Y., (1998, May 1), *There is no strategic OLTP middleware in 1998, Application Integration and Middleware Strategies*, [Online] 5 pages. Available: GartnerGroup Interactive. <http://www.gartner.com> [May 30, 1999]

Schulte and Natis examine the paradox that the developers of new mission-critical business applications face. That is, all of the proven online transaction processing (TP) monitor (OLTP) platforms are soon to be obsolete, but the OLTP platforms of the future are not ready for enterprise usage.

Schulte, R., (1999, February) *Supergroups: IT suites for technology*

integration, Applications Architecture/Frameworks, [Online], 6 pages.
Available: GartnerGroup Interactive. <http://www.gartner.com> [May 30, 1999]

Schulte examines the strategies, technologies, and tools of application suites currently on the market. Information provided supports the theory that a component-based architecture creates a more flexible-computing environment enabling the enterprise to react quickly to changes in the marketplace.

IT Industry Publications

This category of references augment those found in the IT research firm category. Publications listed under this category header provide background information on three-tier architecture, the benefits of using a component-based architecture to develop and deploy cross-enterprise business solutions. These references were used extensively for the distributed architecture and middleware infrastructure analyses. Topics: client/server architecture, component-based application deployment and development, and middleware.

Berson, Alex, (1996, April), *Client/server architecture 2nd edition*, J. Ranade Series on Communications, McGraw Hill.

Berson discusses the advantages of using client/server architecture to develop and deploy business solutions. A three-tier architecture deploys application components (presentation, functional logic, and data) across three tiers of computer platforms: desktop machines, intermediate application servers, and back-end database servers. A key benefit of the three-tier model is each of these application elements is a separate, independent component within the application. This means that each component can be modified or replaced without any of the other components being rewritten or changed. A developer can write an application once and place on middle-tier server accessible by all applications, which brings scalability and reusability into the client/server environment

Edwards, Jeri, (1999, February), *3-tier client/server at work*, John Wiley & Sons.

Edwards describes ten large production 3-tier client/server systems. In interviews with Edwards the architects and the developers of these systems describe how the mission-critical applications were conceived,

designed, built, and deployed across a variety of industries. These case studies while implementing very diverse systems identify several re-occurring patterns and themes about 3-tier that are supported in all selected references. Specifically, the main benefits of 3-tier architecture are scalability, enhanced performance, and efficiency in client/server applications.

Gold-Bernstein, B., (1998, February), *Race to the middle*, Intelligent Enterprise Database Programming & Design, [Online], 15 pages. <http://www.dbpd.com> [March 22, 1999]

Gold-Bernstein discusses the communication modes and mechanisms of a component-based architecture. Focuses on Internet communications, middleware for the middle-tier, and defines the general requirements for an application server platform. Also describes the benefits of three-tier architecture.

Orfali, R., Harkey, D., & Edwards, J., (1997), *The essential client/server guide*, John Wiley & Sons

The authors discuss the advantages of client/server architecture in developing and deploying business solutions. Topics include: object-oriented client/server applications, legacy data access, data warehousing, enterprise data architectures, application software architectures, and methodologies for managing new and existing databases and applications. A very good overview of client/server architecture presented in a straightforward easy to understand format.

Chapter 3. Methods

Using the larger method of literature review, this study is designed to examine the broad area of distributed computing in order to identify selected information relative the issues surrounding cross-enterprise application integration. Enterprise integration is a business problem. It is a new problem brought on by the unprecedented scale, scope, and complexity, of the applications being attempted today. An emergent content

analysis approach, based on grounded theory research design (Strauss and Corbin, 1990) was employed to investigate the issues surrounding this complex phenomenon.

In the initial research design phase a review of the technical literature lead to the definition of a few key generative questions, which helped to focus the selection of literature and guide the investigation efforts (Strauss and Corbin, 1990). Once the basic research questions were defined, the first literature selection was made according to the "principle of theoretical sampling" (p.52). The grounded analysis of the first selection led to the generation of an investigative framework for the examination of distributed computing. As data were gathered, core theoretical concepts were identified through the "process of systematic data collection and data analysis" (p. 23). This iterative process in conjunction with a constant comparison of the data led to the emergence of issues relative to cross-enterprise application integration and of the importance of middleware with respect to those issues.

Data Collection

Data collection focused on the evolution of the middleware solutions that support real-time application integration. Publications were searched from January 1997 to June 1999 in order to describe the larger context and need for this study. That time period was chosen because it parallels the rise in distributed computing with N-tier client/server architecture (an applications development approach that partitions logic across three or more environments: the desktop, one or more application servers, and a database) as the dominant form (BEA, 1999).

As part of data collection, a key word and phrase search using several search engines was conducted to determine the presence of certain words and concepts within materials available in on-line databases, on-line libraries, or referenced in any publications anywhere on the Internet. The key words included distributed computing, three-tier architecture, distributed component platforms, distributed transaction processing, enterprise application integration, electronic commerce and middleware.

The selected published materials searched included:

- Strategic analysis reports, enterprise architecture planning guides, and distributed platform research papers from Gartner Group, Giga**

Information Group, and Forrester Research.

- **White papers from BEA Systems, IBM, Microsoft, and Sun, the leading developers of client/server tools and middleware technologies. Publications from these vendors were searched because they are the current market leaders providing enterprise middleware solutions.**
- **Case studies of companies using these technologies to support their core business processes. These types of case studies were searched to find production systems that use middleware technologies for application integration.**
- **Technical articles and conference position papers on middleware products and strategies that support a high transaction and high security usage pattern required for the integration of enterprise applications.**
- **Business strategy books and journals that define the business problems driving the demand for new middleware solutions.**

Multiple data sources were searched to enhance the validity and reliability of the publication selections through data triangulation methods. (Yin, 1989) Throughout the data collection phase a literature database was maintained to help speed the analysis and provided a mechanism to quickly identify emergent themes. The database was created using tools available in Microsoft's Office suite.

Data Analysis

The data analysis phase of this study was conducted through the use of content analysis, as described by Carley, Weber, and others (as cited by the Colorado State University Online Writing Center Library (CSU-OWL), 1999). This Writing Center supports researchers with online reference materials, and extensive links to other writing and research resources on the Internet.

"As with most other research questions, content analysis begins with identifying research questions and choosing a sample or samples. Once chosen, the text must be coded into manageable content categories. The process of coding is basically one of selective reduction, which is the central idea of content analysis. By reducing the text to categories consisting of word, set of words, or phrases, one can focus on and code for specific words or patterns" (CSU-OWL, 1999).

The specific approach to content analysis utilized in the study was conceptual analysis which focused on selected words and patterns that

were indicative of the research questions under investigation to build a framework of interrelating codes, concepts, and categories. The generation and development of concepts, categories, and propositions was an iterative process as each selected literature sample was examined and compared to others that were collected previously.

This process allowed connections between categories to be established and integrated into a theoretical framework. That process continued until the marginal improvement of the theoretical framework was small according to "the principle of theoretical saturation". Yin (1989) identifies three options:

"(a) Choose a case to fill theoretical categories to extend the emerging theory; and/or (b) choose a case to replicate previous case(s) to test the emerging theory; or, (c) choose a case that is a polar opposite to extend the emerging theory". (p. 53-54)

Once theoretical saturation via literal replication was reached this process concluded. The data gathered in the analysis phase provided information used to construct an overview of the middleware market, descriptions of each main category, and trends in category consolidation. In addition, information relative to trends in distributed component technology, application integration and guidelines for integrating Web-based application into the enterprise was discovered.

Data Presentation

The findings in this study are presented in the form of a list of recommendations that can be used by IT professionals so they can select the best-integrated solution for their needs. The list is intended to provide IT managers with a framework for making strategic investment decisions in middleware technologies.

Chapter 4. Analysis of Data

The Middleware Market

Market evolution has produced a complex array of technologies and has

created so many middleware products that it complicates the understanding and acquisition of solutions. Today, many middleware products are often merged into other product suites, such as application server platforms that deliver value-added functionality. The market is evolving toward higher value solutions that manage enterprise scale processes more effectively and middleware is increasingly viewed as mission-critical infrastructure.

In this report the middleware market is segmented into six functional areas as revealed through data collection and analysis in the literature review. However, other publications and research groups may segment this market a little differently. The primary categories are transaction processing (TP) monitors, message-oriented middleware (MOM), object request brokers (ORB), data access middleware, application servers, and object transaction monitors (OTM).

Transaction processing (TP) monitors: this category of software provides process, transaction, and communications management. Services include distributed transaction monitoring, management, load balancing and fail-over across multiple data sources. Provides a framework for running server applications and APIs for development and runtime of online transaction processing (OLTP) applications. A TP monitor is a proven enabler for high-volume OLTP applications.

Message-oriented middleware: message-oriented middleware (MOM) handles the process of distributing data and control through the exchange of records known as messages. "Messaging" encompasses several levels of technology that range from simple synchronous/asynchronous messaging to IP multicast. This category is often used to integrate distributed applications through publish and subscribe, message queue, and/or data transformation. These models will be addressed in greater detail in a later section.

Object request brokers: an object request broker (ORB) provides an infrastructure that facilitates cross-platform communication between distributed objects and client applets. ORB is defined in the Common Object Request Broker Architecture (CORBA) specification. This specification defines the way objects are created, dispatched, invoked, and how they communicate.

Data access middleware: products in this category provide efficient and rapid access to heterogeneous data sources via SQL requests in relational

databases. Also invokes stored procedures. Vendor supplied APIs gives full access to all database (DB) features and are best for high-volume online transaction processing (OLTP).

Application servers: an application server is used to host the middle-tier business logic of a multi-tier client/server application environment and automates some of the more complex features of multi-tier computing. Application servers manage and recycle scarce system resources, such as processes, threads, memory, database connections, and network sessions on behalf of the applications. They are the result of the consolidation of several middleware categories that combines ORBs, messaging, and database access into a value-added platform.

Object transaction monitors (OTM): This category is the next generation TP monitor based on the CORBA object standard from OMG, Enterprise JavaBeans from JavaSoft, or DCOM from Microsoft. OTMs combine the functions of TP monitors, ORBs and MOM into a value-added transaction processing middleware system. The distributed object infrastructure provides TP monitors with a myriad of standard middleware services -- including metadata, data transformation and movement, workflow and application integration.

The middleware market can be further segmented into three broader categorizations ***data integration:*** products enabling access and usage of data; ***process integration:*** products linking distributed application components or processes into a unified application view and ***distributed components:*** a subset of process integration. Currently, a "war" is being fought in the distributed component arena to define the prevailing network architecture for the next generation of distributed Internet, Intranet, and Extranet applications.

According to (Gilpin, 1998),

"Control of the standards that define the middleware drives another battlefield, although an uneasy truce has been declared. Most enterprises are in a situation that demands that they use both Windows NT and UNIX servers, which requires them to support multiple component and middleware standards. ... 50 percent of the Fortune 500 companies have deployed some combination of COM, CORBA, and Enterprise JavaBean (EJB) technology, with interoperability required as a part of the solution. But despite this acceptance of multiple standards, the promoters of each standard continue to strive to maximize the amount of territory they

occupy" (p.2).

Today, the middleware market continues to evolve toward solutions that provide the infrastructure for distributed component applications. However, regardless of what the future might hold, middleware is already viewed as an essential component of any enterprise client/server application.

Distributed Component Architecture

Application architecture dictates the way in which an application is created and how its components are distributed across systems. Most application programs are composed of three fundamental types of application components.

A presentation component contains the logic that presents information to the end-user working at a terminal or workstation. The presentation logic generally provides a menu of options to allow the user to navigate through the different parts of the application, and it manipulates the input and output on the display.

A business component contains the application logic, which dictates how the business functions and processes are performed by the application. These functions and processes are invoked either by the end-user when he selects an option, or by other business functions that generally perform some type of data manipulation.

A data access component contains the logic, which interfaces a data storage system such as database systems, file systems, or some other type of external data source. Data access functions are usually invoked by business functions.

One-Tier

In the traditional mainframe environment, applications are not decomposable into their fundamental component types because it's difficult to separate the component logic. Instead, all three components are tightly bound in an integrated single executable program, which runs on a single machine. These applications run either as batch jobs or as time-shared online applications driven by users sitting at character-based terminals (Berson, 1996).

This "one-tier" architecture is based on a comprehensive environment

associated with the hardware platform. The one-tier model provides a number of significant advantages. Since the application is centralized in a single well-known environment, it is easy to manage, control, and secure. The online transaction processing (OLTP) environments provided in mainframe systems are dependable and reliable and they support 24 hour by 365-day application availability. They can support high transaction volumes and large numbers of users. An online transaction processing (OLTP) application architecture is still the cornerstone of most mission-critical application systems in use today. (Berson, 1996)

However, there are also a number of significant disadvantages associated with the centralized approach. Since one-tier applications are confined to a single processor, scalability can be a costly. If the system becomes overloaded, your only recourse is to upgrade to a larger machine. There is a direct correlation between application size and system cost. As the application gets larger the scalability price get higher, and it is the multi-million dollar price tag associated with large mainframe systems that was one of the key drivers that forced companies to seek alternate solutions. (Anderson, 1998) Additionally, since all of the application components are bound into a single executable it is difficult to make modifications to the application systems in response to the changing business requirements. Also, because the application logic is only accessible through the integrated presentation logic it's very difficult to interface with other systems or to access the application functions from outside the application system (Orfali, Harkey and Edwards, 1997). Lack of flexibility is probably the most significant disadvantage associated with this approach.

Two-Tier

In two-tier client/server architecture an application is split into two parts and divides the processing between a desktop workstation and a server machine. Communications between the graphical user interface (GUI) and the database is accomplished using the SQL-based proprietary communications protocol provided by the database vendor. This type of client/server architecture is also referred to as "fat client" because most of the application logic runs on the desktop. (Edwards, 1999, p.11)

The GUI development tools allow much more rapid development and deployment of applications. By off-loading much of the application processing to the desktop workstations, the server systems do not need to be as large. The hardware-independent database systems allow easy

portability between systems, and using Microsoft's Open Data Base Connectivity (ODBC) interface, the GUI tools provide a level of independence to the relational database vendors. ODBC also provides easy access to relational databases from the most popular reporting and spreadsheet tools.

However, the two-tier approach only works effectively when you are developing simple applications, accessing a single relational database, and supporting a small user base. As applications become more complex, in terms of business algorithms processed, number of databases accessed, or number of users supported, this model starts to fall short. Without the tight security controls provided by a centralized environment, each client application must enforce its own security process. Since the relational databases can be accessed from a number of different clients, each database must also enforce its own security process.

Additionally, since the GUI development tools are dependent on the database vendor's proprietary SQL-base communications protocol in order to attain interoperability with existing environments, point-to-point connections must be manually constructed and maintained. The client applications become increasingly more complex and harder to maintain

Three-Tier

The three-tier client/server architecture provides an environment that supports all the benefits of both the centralized model and the two-tier approach, and also supports the goals of a flexible architecture. The three-tier application model splits an application into its three logical component types, presentation logic, business logic, and data access logic. There may be any number of each of the component types within an application. Any number of application systems can share application components. A distributed computing infrastructure provides location, security, and communication services for the application components (Edwards, 1999). Figure 1 illustrates a distributed component-based deployment architecture.

Benefits of Distributed Components

A component-based application offers significant advantages over traditional mainframe based applications. (Gold-Bernstein, 1998) The true measure of an enterprise application system often comes down to system

scalability and total throughput. How many users can concurrently use the system? How many transactions can be processed per second? Multi-tier applications provide a number of significant advantages over traditional client/server architectures, including improvements in scalability, performance, reliability, manageability, reusability, and flexibility.

Figure 1: Distributed Component Architecture



Object Reuse: each module is actually a shareable, reusable object that can be included in other application systems. This plug and play versatility is useful when IT needs to support different, but related parts of the business. For example, a strategic application within a sales department should be able to view inventory information and interface with the order entry system. Therefore instead of writing the same function or service in every application, developers can write the application once and place it on an intermediate middle-tier server accessible by all applications which brings scalability and reusability into client/server environments.

Ease of System Maintenance: since application functions are isolated within application objects, application logic can be modified much more easily. For example, one function performed by a financial application is to project post-tax earnings. This function changes periodically as the tax regulation changes. Normally, this requires significant modifications to the entire financial application. By isolating these business rules into an independent business object, it can be changed to match the tax regulations without impacting the rest of the application. Each application component is

developed using the best tool for the job. The application components can be deployed across one or more physical systems. The application components communicate with each other using an abstract interface that hides the underlying function performed by the component.

Abstract Interface: the object-oriented concepts of encapsulation and abstraction are fundamental to the three-tiered architecture and to application flexibility. Each application component can be viewed as an encapsulated object -- a data structure with a set of operations or methods that can be used to manipulate the data. The data within the object can only be manipulated by using one of the defined operations. The operations are invoked using an abstract interface. The abstract interface identifies the operation to be performed and defines the input and output parameters which are required to perform the operation.

Effective Use of Data and Networks: application logic is no longer tied directly to the database structures or a particular DBMS. Individual application objects work with their own encapsulated data structures. When application objects communicate, they only need to send the data parameters as specified in the abstract interface rather than entire database records, thereby reducing network traffic. The data access objects are the only application components that interface directly with the databases. The object independence of the component-based model provides IT departments with the ability to better react to either business or technological changes.

Faster Application Development: with two-tier methodologies, each programmer must develop all aspects of an application, including presentation, business, and data access logic. This approach does not exploit the fact that most programmers excel in certain tasks and not in others, and that they are more productive when they are specialized. Using a three-tier model, programmers who have excellent user interface skills can concentrate on developing powerful presentation components. Database analysts can focus their development efforts on optimizing data access while business analysts concentrate on developing business algorithms.

Platform Independence: the various application components can be written using different languages and programming environments. Therefore the presentation component can be built using a powerful GUI development tool, while the business logic can be written in a business language such as COBOL, and the data access components can be written in C++ with

embedded SQL. This allows the custom application to be incorporated with "off-the-shelf" components.

Although standard middleware allows seamless communications between clients and server components, as well as between other server components, it does not guarantee plug-and-play capability. What does that mean? According to (Edwards, 1999),

"While mixing and matching components sounds wonderful — and is wonderful. Buying component suites that use the same middleware -- even loosely coupled dynamic messaging mechanisms such as queues and publish and subscribe -- does not mean out-of-the-box interoperability because there are no communication standards. Components must know how to access each other semantically. One frequently used mechanism... is to develop a new application that bolts them together...In all cases, someone must provide the glue" (p.13).

Communication Methods

There are three basic communication modes for distributed applications: conversational mode, remote invocation, and messaging. Conversational mode is a synchronous communication, both parties actively engage in the conversation. In this mode the state of a query and transactions are maintained by the database. The problem with conversational mode is that it requires a separate connection to the database for each process, and therefore does not scale well. Mission-critical enterprise solutions require scalability because of the high volume of users and unpredictable workloads.

In remote invocation an application invokes the services of another segment of application code on another computer. Remote invocation is a higher performance mechanism that extends common structured programming techniques to a distributed architecture. This mode is widely used multi-tier and thin-client distributed applications through four mechanisms RPCs, OMG's CORBA ORBs, Microsoft's COM, and Sun's Enterprise JavaBeans. (Gold-Bernstein, 1998) (Gilpin, 1998, December, 30)

CORBA

CORBA is a set of specifications published by the Object Management Group (OMG), an industry consortium of 700+ companies whose mission is to define a set of cross-platform interoperability interfaces for object-oriented software. Figure 2 illustrates the CORBA architecture. These

specifications define the way objects are created, dispatched, invoked and how they communicate with each other. A CORBA compliant ORB therefore, is middleware that allows a client object to issue a request against a service object. A key component of the ORB technology is IDL, the specification language that documents the interface between the clients and servers (OMG, 1999) defines the following components for the CORBA specification:

Object Implementation: this defines operations that implement a CORBA IDL interface. Object implementations can be written in a variety of languages including C, C++, Java, and Smalltalk.

Client: this is the program entity that invokes an operation on object implementation. Accessing the services of a remote object should be transparent. The remaining components help to support this level of transparency.

Object Request Broker (ORB): the ORB provides a mechanism for transparently communicating client requests to target object implementations. The ORB simplifies distributed programming by de-coupling the client from the details of the method invocations. This makes client requests appear to be local procedure calls. When a client invokes an operation, the ORB is responsible for finding the object implementation, delivering the request to the object, and returning any response to the caller.

ORB Interface: an ORB is a logical entity that may be implemented in various ways (such as one or more processes or a set of libraries). To de-couple applications from implementation details, the CORBA specification defines an abstract interface for an ORB. This interface provides various helper functions such as converting object references to strings and vice versa, and creating argument lists for requests made through the dynamic invocation interface.

CORBA IDL stubs and skeletons: CORBA IDL stubs and skeletons serve as the "glue" between the client and server applications, respectively, and the ORB. A CORBA IDL compiler automates the transformation between CORBA IDL definitions and the target programming language. The use of a compiler reduces the potential for inconsistencies between client stubs and server skeletons and increases opportunities for automated compiler optimizations.

Dynamic Invocation Interface (DII): this interface allows a client to directly access the underlying request mechanisms provided by an ORB. Applications use the DII to dynamically issue requests to objects without requiring IDL interface-specific stubs to be linked in. Unlike IDL stubs (which only allow RPC-style requests), the DII also allows clients to make non-blocking *deferred synchronous* (separate send and receive operations) and *one-way* (send only) calls.

Dynamic Skeleton Interface (DSI): this is the server side's analogue to the client side's DII. The DSI allows an ORB to deliver requests to an object implementation that does not have compile-time knowledge of the type of that the object it is implementing.

Object Adapter: this assists the ORB with delivering requests to the object and with activating the object. More importantly, an object adapter associates object implementations with the ORB. Object adapters can be specialized to provide support for certain object implementation styles (such as OODB object adapters for persistence and library object adapters for non-remote objects).

Figure 2: CORBA Architecture



COM

The Component Object Model (COM), the core object technology within Microsoft's ActiveX™, refers to both a specification and implementation that provides a framework for integrating components. COM defines an application-programming interface (API) to allow creation of components for use in integrating custom applications or to allow diverse components to interact. In today's operating systems, processes are shielded from each

other. A client that needs to communicate with a component in another process cannot call the component directly, but has to use some form of inter-process communication provided by the operating system. COM provides this communication transparently. It intercepts calls from the client and forwards them to the component in another process.

Distributed Component Object Model (DCOM) is the distributed extension to COM that builds an object remote procedure call layer on top of DCE RPC to support remote objects. Figure 3 illustrates Microsoft's COM/DCOM architecture. A COM server can create object instances of multiple object classes. While COM processes can run on the same machine but in different address spaces, the DCOM extension allows processes to be distributed across a network. (Microsoft, 1997).

When client and component reside on different machines, DCOM simply replaces the local inter-process communication with a network protocol. The COM run-time provides object-oriented services to clients and components and uses RPC and the security provider to generate standard network packets that conform to the DCOM wire-protocol standard

Figure 3: COM/DCOM Architecture



COM and DCOM are most mature on Windows platforms. A component marketplace with a wide selection of COM-compliant objects has developed for Windows. However, the step from a Windows platform-specific COM to distributed multi-platform support is quite significant.

Enterprise JavaBeans (EJB)

The Enterprise JavaBeans architecture defines a standard model for Java application servers to support "Write Once, Run Anywhere™" (WORA) portability and extends the JavaBeans component model to support server

components. The Java virtual machine (JVM) is the mechanism that allows a Java application to run on any operating system. However, Java server components require additional services that are not supplied directly by the JVM. These services are supplied either by an application server or by a distributed object infrastructure, such as CORBA or DCOM.

Java technology clients invoke the application using the native Java Remote Method Invocation (RMI) interface. RMI requests currently are transferred using the Java Remote Method Protocol (JRMP). In the future, RMI will be extended to support the industry-standard Internet InterORB Protocol (IIOP). Native language clients can invoke the application using CORBA IDL running over IIOP or a COM/CORBA inter-networking service running over IIOP. The RMI client proxy could also be rendered as an ActiveX control to provide easy integration with any Windows application. Browsers can invoke the application through a servlet running on the HTTP server. The browser communicates with the servlet using HTTP, and the servlet communicates with the application using RMI. Figure 4 illustrates EJB architecture. An Enterprise JavaBeans-compliant application server supports complete portability. For example, TP monitors, CORBA-based systems, COM runtime systems, DBMS, Web server systems, or other server-based runtime systems can be adapted to support portable Enterprise JavaBeans components. (The Source for Java™ Technology, 1999)

The EJB container implements the management and control services for one or more classes of Enterprise JavaBean objects. Additionally, the container provides lifecycle management, implicit transaction control, persistence management, transparent distribution services, and security services on behalf of the enterprise bean.

Figure 4: Enterprise JavaBean Architecture



Message-Oriented Middleware

Message-oriented middleware (MOM) handles the process of distributing data and control through the exchange of records known as messages. "Messaging" encompasses several levels of technology that range from simple synchronous/asynchronous messaging to IP multicast. (Edwards, 1999) (Gold-Bernstein, 1998) In this category there is no common source code or set of specifications and there are more than a dozen MOM products that span a wide range of functionality driven by diverse communication models.

***Message Passing:* at the simplest level is a direct program-to-program connection-oriented communication model. Using message passing, an application request is sent in the form of a message from one program directly to another. The communications mechanism can be either synchronous (i.e. the sender is blocked until the receiver sends a message back), or asynchronous employing a polling model, or through callback routines. Message passing is suitable for event-driven applications, but because of the logical connection requirement it is not suitable for loosely coupled, time independent applications.**

***Message queuing:* at the next level is an indirect program-to-program connectionless communication model that allows programs to communicate via message queues rather than connecting directly. In this model the queue is opened, messages are then written in, after being read by the other program the queue is then closed. A queue manager collaborates with other queue managers throughout the network to control routing. The queue manager also handles other control functions such as message acknowledgment, prioritization and load balancing. Queuing products support guaranteed message delivery, triggers (a feature that allows**

applications to be active only when work is done), and sessions. These features provide reliable performance, avoid unnecessary consumption of resources, and reduce the number of required connections, respectively. Overall, MOM is a flexible attractive model for the development of event-driven enterprise distributed applications.

Publish & subscribe: this model has evolved from trading applications in the financial industry. Once considered a niche product they are now widely used by customers that need maximum messaging throughput in situations that are less peer-to-peer oriented than the solutions above. In this model applications publish information on the network to subscribers on topics of interest. A classic example is a stock ticker, where there are quotes on a lot of different symbols, and various traders wanting up-to-the-moment information, each on a particular subset of all the available symbols. An advantage of this communication paradigm is that it enables the development of flexible business systems since the publisher & subscriber don't need to maintain any "state" information. This means that the systems can be dynamically reconfigured without interruption of operation and allows for efficient cross-enterprise application integration.

IP Multicast: a new standard from the Internet Engineering Task Force (IETF). The standard extends the way IP works by enabling information to be efficiently communicated in a publish & subscribe paradigm over a WAN. An IP packet can be transmitted once by the sender, and routed to multiple recipients. This approach can be optionally chosen as a particularly efficient way of doing publish & subscribe over a TCP/IP network. It is also well suited to audio or video streaming applications. Message-oriented middleware is also being merged into both ORB and Java technology, expanding the range of application types that can be built using the object paradigm.

Middleware Infrastructure

"The middleware infrastructure itself is a component architecture" (Gold-Bernstein, 1998, p.5). Middleware components provide the services that allow the application components to be transparently distributed across any number of physical systems, a concept often referred to as partitioning. (Eckerson, 1996) Partitioning of application components introduces a wide range of new technical issues. In the two-tiered environment, communications are accomplished using the DBMS connectivity environment. The developer must identify the location of the database

system within the application, and the client application and/or the database system performs the necessary security checks. On the other hand, in the three-tiered environment, communications are performed through an abstract interface. Server location, security checks and communications should be handled dynamically by the middleware infrastructure. This allows dynamic re-configuration of distributed application to accommodate more users, increased workloads, or unexpected hardware failures

The critical component for the development and deployment of enterprise mission-critical three-tier client/server applications is a middleware infrastructure that enables application partitioning, transparency, security, scalability, reliability, availability, and manageability. (Eckerson, 1996). In a Forrester report (Schadler, Woodring, Overby, and Walker, 1998) describe the requirements to achieve those capabilities. They state, "...Forrester believes that companies must extend their component platform to include a set of services that enables a new loosely coupled integration strategy that we call "federated integration"(p.6). Both (Gilpin, 1999) and (Schadler, et al., 1998) identify four main categories of services that must be provided:

- ***Application Services:*** the business logic needed to implement new processes or enhance existing applications. A standard set of manageability services that ensures the continued integrity, or correctness, of an application include security, concurrency and serialization control, and server management.
- ***Connectivity Services:*** basic middleware services -- network connectivity and communications protocols. (ORB, RPC, DCOM, HTTP, IIOP, messaging, etc.). Connection services include both data-level and event-level integration.
- ***Management Services:*** robust administration tools that provide event-notification, exception-based alerts, system testing, and automated performance tuning.
- ***Transformation Services:*** Data and events needed to be translated so they can be understood by all applications.

Application Servers

The middleware market is consolidating categories of solutions into application server platforms. Object request brokers (ORBs), message-oriented middleware (MOM), transaction management and database access middleware solutions are being repacked into application servers. This is

being done in conjunction with the addition of more deployment, runtime and management functionality. Companies considering application servers can look to the products and technologies used on Wall Street as an indicator of what elements are required to comprise a successful application server solution.

At the April 1998 Wall Street Middleware Working Group inaugural meeting Mike Gilpin, Vice President of Giga Information Group and event moderator stated

"Until now, the movement to application servers on Wall Street has been a matter of leading-edge investment banks building their own middle-tier infrastructure. These Wall Street users now understand what it takes for middleware solutions to be highly scalable and to be able to handle real business problems. Offering this insight to vendors ... building the next generation of middleware products will help ensure that products will provide an ideal fit for the needs of the financial industry". (Wall Street Middleware Working Group (WSMWG), p.1)

Application servers are now emerging into the mainstream as products because of the lessons learned by this group of first generation distributed application developers. In the future these companies plan to rely more on vendors for integration solutions rather than doing their own system integration. Also, they plan to choose object request broker and messaging technology for a high performance scalable solutions, and language independent connectivity solutions (i.e., CORBA) for legacy system integration ability.

At an October advanced technology conference for executives (Waters Enterprise Group, 1998) Chris Keene, president of Persistence Software and cofounder of the Wall Street Middleware Working Group gave a presentation on transactional application servers based on CORBA technology. He indicated that the Middleware Group as a whole is strategically committed to CORBA technology because of its ability to provide a ubiquitous solution for multi-platform distributed application connectivity and runtime support. According to Keene,

"Although, members in the Group have used messaging technology as well, they are looking to CORBA, and new emerging techniques for CORBA as the best combined messaging solution going forward, for supporting their needs for building advanced distributed architectures for the middle tier. This layer provides the "glue" that enables the front office and back-office components of trading systems to be flexibly linked on a

common infrastructure for securities trading" (Keene, 1998, p.3).

An application server automates some of the more complex features of multi-tier computing. It manages and recycles scarce system resources, such as processes, threads, memory, database connections, and network sessions on behalf of the applications. Some of the more sophisticated application servers offer load-balancing services that can distribute application processing across multiple systems. Additionally, an application server provides access to infrastructure services, such as naming, directory, transactions, persistence, and security. However, until recently, most applications servers were bundled into enterprise application development platforms which include complete GUI development tools, database connectivity components, and deployment facilities. (Gilpin, 1999) Driven by the growth of the Web as an information and commerce platform, application server products are now heavily biased toward the Internet. They leverage Web protocols such as HTML, and HTTP and use the Web browser for client access. (Costa, 1999, January 6)

For today's enterprise applications, flexible resource use is a key to success. Building component-based applications enables the use of the latest application server technology to meet individual business challenges. For most resource-intensive applications, dedicated TP monitors offer the highest performance. Developing in Java will open a wide range of platform choices for application deployment and most new technology is converging around Internet standards. However, none of the current products in any group provides a perfect solution. Companies should choose application servers that are oriented toward the types of applications they will need most often, or reflect the largest future application investment.

Trends in Distributed Component Technology

Companies in all industries are utilizing middleware to improve customer value; speed cycle times, and create a more flexible-computing environment, enabling the enterprise to react quickly to changes in the marketplace. (Enslow and Schulte, 1998) Today, electronic commerce is driving demand for middleware solutions to enable the seamless integration of disparate systems as enterprises are forming Internet-based trading groups, integrated supply chains, and virtual companies to create a unified value chain (Gilpin, 1998) (Brown, 1999). Distributed component technology is evolving to influence application architectures and the requirements placed on middleware solutions. (Gilpin, 1998) identifies technical trends

currently driving the middleware market:

All categories of middleware technology are being driven to deliver greater customer value, moving away from low-level technology to higher-level solutions. This requires the integration of multiple related technologies like messaging, CORBA object request brokers, message brokers, Java-based solutions and development tools into value-added platforms like application servers.

Distributed component technology is evolving to influence application architectures and the requirements for middleware solutions. Microsoft COM is moving to DCOM. JavaBeans are being enhanced, and Enterprise JavaBeans is bringing Java components into mainstream server usage. The demand for COM and CORBA interoperability solutions is increasing as more organizations adopt both standards for their enterprise application architectures and in the longer-term the most successful integration solutions will be those that leverage distributed component architecture.

The main business drivers for middleware are (1) Enterprises are obtaining greater value through increased application integration of the supply chain. This value is passed along to customers in the form of improved service, higher quality products and services that better meet requirements, and lower costs of acquisition and ownership (Brown, 1999), (Dec, et al., 1998) and (Papows, 1998). (2) Ubiquitous network connectivity and device intelligence are creating new business opportunities, such as new services in telecommunications. This is driving demand for cross-enterprise integration solutions to connect widely distributed heterogeneous architectures. (Weil and Broadbent, 1998) (3) An increasing number of enterprises are engaging in electronic commerce, driving demand for middleware solutions that enable enterprise infrastructures to function as a unified whole and is also driving the demand for solutions that manage enterprise-scale processes more effectively. (Freidheim, 1999) (Brown, 1999).

Middleware solutions are evolving away from low level technology to higher level solutions. The results are higher level value-added solutions like application server platforms and enterprises are obtaining greater value through increased application integration. The increasing popularity of e-commerce initiatives is driving demand for middleware solutions that open systems to Internet, Intranet and Extranet usage.

Application Integration Solutions

Enterprise integration solutions that combine new custom development, packaged solutions and legacy applications into new higher functioning combinations are the fastest-growing middleware category in 1999. While solutions may vary, a common vision of their constituent elements is emerging. The use of development tools and runtime environments reduce configuration and customization effort, speed time-to-market and reduce total cost of ownership, compared to custom programming. Because integration solutions are often driven by IT business initiatives like e-commerce or supply chain management, their economic impact can be significant. However, all the solutions require some systems integration effort

Year 2000 (Y2K) remediation has increased the purchase of "off-the-shelf" packaged solutions and the need connect legacy systems to the Web is accelerating the demand for application integration solutions. Integration technologies are being used by both systems integrators and customers to connect multiple enterprise systems (packaged solutions, legacy and new applications) into combined value added solutions

Integration solutions rely upon a variety of infrastructure approaches, each appropriate to meet different integration needs. For example, message-based solutions use store and forward, or publish and subscribe paradigms, enabling near real-time service while not requiring connected applications to be online at the same time. One benefit of this approach is that the overall system can be dynamically re-configured, new clients or services added without interruption of operations. While, object-based middleware require component-based application architecture and their purpose is to enable communication between distributed objects. Object request brokers let objects transparently make requests to --and receive responses from -- other objects locally or remotely.

Web Integration

Most early data-driven Web applications were developed using stand-alone databases in large part because Web integration was extremely difficult. Not only were the tools immature and unreliable, but they offered little functionality beyond database query via SQL and ODBC. Today, Internet application integration requires a middleware architecture that unites the Web, client/server, and legacy system resources into a unified platform capable of delivering real business value. (Gilpin, 1999, May 10) However, while the current technologies allow developers to create

applications and business processes that span multiple systems, the integration process is still extremely complex. That is why IT departments are demanding -- and vendors are starting to deliver -- tools that take some of the complexity out of application integration.

Middleware vendors are starting to bundle their product offerings into a single package with an easy-to-use interface that doesn't require programming knowledge. However, while web integration is easier (Costa, 1999, January 6) identifies several factors that organizations should consider carefully when tying web-based applications directly into enterprise systems. (1) *Security*: highly sensitive data will require digital certificate protection, which is still difficult to implement and not widely supported among web development tools. (2) *Scalability*: even though legacy applications may perform well in their existing environment they may not scale well enough to support the unpredictable work-loads generated by a Web application. This type of limitation can only be addressed by redeveloping or replacing applications so that they can be better leveraged in future deployments. (3) *Performance*: depending on the location of the applications and data, as well as the method of integration, real-time queries of the data may not offer acceptable performance. Particularly, in cases where queries would have to be performed over a WAN or where data would have to go through multiple transformations before the results are returned to the Web client.

Chapter 5. Conclusions

Middleware is an essential component of distributed computing because it provides the services and protocols necessary to integrate disparate systems and architectures into a seamless whole. Companies in all industries are utilizing middleware to improve customer value; speed cycle times, and create a more flexible computing environment that enables the enterprise to react quickly to changes in the marketplace while maintaining the viability of older legacy systems. A middleware infrastructure can help to facilitate corporate restructuring by eliminating the duplication of business functions and applications resulting in improved overall efficiency.

As demonstrated in this research study, middleware industry consolidation is reducing the number of categories to consider in most cases and making it easier to achieve business strategy alignment. Although, there has been a great deal of consolidation in the middleware industry that has led to an

increased emphasis on higher-level, higher-value solutions like application servers, all solutions still require some system integration effort. Additionally, there are certain trade-offs that must be made when choosing among the middleware solutions for integration of new and legacy applications, and with PC and Web clients. However, it is possible to develop a framework to guide decisions about middleware according to individual technical requirements. The following list of recommendations are meant to simplify choices for IT professionals and to provide IT managers a perspective to help ease the decision making process.

Recommendations

- ***Devise a middleware strategy:*** IT managers must define a strategic application technical architecture for their companies, then define the middleware solutions that are consistent with that architecture.
- ***Define requirements first:*** IT managers should develop a general approach to guide decisions about middleware based on the strategic business objectives of the company. The first step in deciding among the various middleware products is to determine exactly what problems need to be solved. For example, if client access or application integration is required the choice is between MOM and ORB. In cases where continuous availability and transaction integrity is required the choice is an application/transaction server with services that support virtual networking, fault-tolerance dynamic configuration management, load balancing and fail-over. IT managers should choose middleware solutions that are oriented toward the types of applications they will need most often, or reflect the largest future application investment.
- ***Wait for the server component market to mature:*** One of the promises of distributed component technology is to provide an environment in which customized business solutions can be assembled from a set of "off-the-shelf" business objects. While CORBA, Enterprise JavaBeans, and COM promise to dominate architecture planning, the market for server-side components is still very young. As more organizations adopt a distributed component architecture, the market is likely to mature rapidly. Some e-commerce vendors are beginning to supply individual application functions, such as a shopping cart and a credit validation service, as customizable components.
- ***Shared servers:*** In order to achieve the most benefit from the multi-tier architecture, server components should be implemented as shared

servers. Highly scalable shared servers need to support concurrent users, participate in business transactions, and efficiently share scarce system resources, such as threads, processes, memory, database connections, and network connections. Additionally, a shared server needs to enforce security policies

- ***Independent standards decisions: COM, CORBA, or Enterprise JavaBeans*** There are advantages and disadvantages in all three architectures. COM is built on a proven desktop component architecture and COM-based applications are robust and perform well. The key issue is risk of a new technology. COM is still unproved as an enterprise technology. In contrast, CORBA and EJB are more complete and well-define architectures and provide better solutions for heterogeneous environments. The disadvantages of both are their complexity and variation in vendor implementations.
- ***Use only one CORBA or Enterprise JavaBean vendor:*** CORBA and EJB are a set of specifications with few implementation details. Each vendor implements the standard and then adds proprietary features which can cause interoperability and portability issues. Both CORBA and EJB offer multi-language support and are platform independent. However, CORBA as a standard for integrating distributed objects offers advantage in services, platform and support tools, maturity, and overall integrity. OMG has defined a COM - CORBA specification, which allows distributed applications to be developed using both. COM is a Microsoft proprietary technology and not a true multi-platform solution.
- ***COM and DCOM:*** Microsoft is the primary holdout against interoperability with both EJB and CORBA technology. Although Microsoft Transaction Server (MTS) could be adapted to support their components, Microsoft is not likely to make the effort. Portability would undermine Microsoft's message of tight integration based on the Windows NT platform. Microsoft is pushing developers to build component-based application systems based on COM. MTS provides a container system for COM server components, providing transactional and security services. COM+, the next generation of MTS, will provide a few additional capabilities, such as dynamic load-balancing and queued request processing.
- ***Consider message-oriented middleware.*** IT managers should consider using messaging where an application is mission critical and has requirements for both data movement and event processing. However, if your objective is real-time application integration, a solution using message brokers or object request brokers would be a better choice.

- ***Real-time application integration solutions:*** Look at the products and technologies used on Wall Street for an indicator of what elements comprise a successful application server solution.
- ***Java and Java Script:*** Java adoption is explosive. The Web application server market will consolidate around Java and Java Script as the standards for accessing application server resources.
- ***Consider an "off-the-shelf" application suite:*** Solutions for integrating new, packaged and legacy applications into a high-level aggregate are the fastest-growing middleware category. All solutions require some systems integration. However, their tools, pre-built adapters and runtime environments reduce customization and configuration effort, speed time to market and reduce cost of ownership, compared to developing custom code.
- ***Expect impacts from DCOM when Window 2000 is released:*** The most important event for DCOM will be the full commercial release of Windows 2000. It will include an advanced set of middleware-oriented services called Active Directory that should make DCOM much more powerful and easier to use. However, it is still too early to tell future versions of DCOM will compete with CORBA and the growing market for EJB.

Appendix A

Definitions

The following definitions are provided in this study as reference to ensure uniformity and understanding throughout the paper. The definitions for the terms defined below come from a BEA Systems Inc. publication "Speaking Middleware" unless otherwise referenced. BEA is one of the top vendors of enterprise middleware solutions and currently has 55% of the transaction processing (TP) monitor market.

Active-X: A set of client technologies from Microsoft that enables software components to interact with one another in a networked environment, regardless of the language in which the component were created. Active-X is built on the Component Object Model (COM) and includes OLE functionality. Also see Component Object Model (COM) and OLE.

Application programming interface (API): A set of code that enables a

developer to initiate and complete client/server requests within an application.

Availability: Features of transaction systems that contribute to the smooth, continued operations in the presence of failures.

Client/server: A relationship between two processes in which one makes a request to the other. Involves two or more computers sharing tasks related to a complete application. The client is defined as the requester of services and a server is defined as the provider of services (IEEE, 1990)

Common object request broker (CORBA): Defined by the Object Management Group (OMG), it defines the components of an open object bus and how those components inter-operate. It also specifies an extensive set of services for creating and deleting objects, accessing them by name, storing them in persistent stores, externalizing their state, and defining ad hoc relationships between them.

Component object model (COM): The object model used on Microsoft platforms to define objects and how they inter-operate. COM is different from CORBA in many ways. For example, there are differences in the mechanisms that reference objects and in the processes that create them. Also See Common object request broker (CORBA).

Distributed transaction processing: A middleware approach that extends the capabilities of traditional OLTP systems across platforms and networks. Designed to provide the reliability and high availability needed for high-volume applications.

Electronic commerce: The process of completing a business cycle — from product introduction, to pricing, delivery, and acceptance, through the exchange of assets — electronically.

Failover: Fault management techniques in which the processing functions of a failed component (hardware or software process) are assumed by another component — generally a replicated instance of the failed component. A transaction processing system may implement failover techniques by maintaining replicated component instances, detecting failed instances, and routing requests and messages to only those component instances that are "alive" and responding normally.

Instance: A specific object within a class. For example "Microsoft Word" is a

specific instance of the class "word processors".

Interface: A set of operations and attributes that can be performed on an object. An interface is defined by using Object Management Group (OMG) interface definition language (IDL) statements to create an interface definition.

Java: An object-oriented programming language modeled after C++. It is designed to be small, simple, and portable across platforms and operating systems.

Load balancing: The ability of a system to ensure maximum application throughput by automatically finding the most available server for a request, then sending the request to that server, or the server's queue, for processing.

Message: Information sent between applications (processes). Messages can include data, program instructions, or both.

Message-oriented middleware (MOM): A middleware approach for implementing N-tier client/server that combines message queuing and message passing in real-time. Benefits include supporting asynchronous communications, providing inherent parallelism, and scaling across thousands of network nodes.

Middleware: Software that runs within or between parts of a distributed client/server application. Middleware approaches for implementing N-tier client/server include remote procedure call, distributed transaction processing, object components, and message-oriented middleware.

N-tier client/server: An application development approach that partitions application logic across three or more environments: the desktop, one or more application servers, and one or more database servers. The main advantage of n-tier client/server is that it extends the benefits of client/server to the enterprise level. Other advantages include added manageability, scalability, security, and higher performance.

Object Management Group (OMG): A consortium of more than 700+ companies that define the Common Object Request Broker Architecture (CORBA).

Object request broker (ORB): An object bus that lets components inter-

operate across address spaces, languages, operating systems, and networks; it provides a mechanism that lets components exchange metadata and locate each other.

Object transaction manager (OTM): A new type of middleware that provides both a fast, reliable communications pipe and a management platform for objects, including all the methods that compose their interfaces as well as their state.

On-line transaction processing (OLTP): The area of business computing which involves mission-critical business transactions that are processed in real-time. These systems require high volume throughput and rapid response times. Examples of OLTP applications are point-of-sale (POS) applications in retail stores and automated teller machine (ATM) banking.

Publish and subscribe: A form of distributed data processing driven by business events. Middleware is used to trigger a series of processing steps across multiple applications in response to pre-specified events such as electronic orders or stock price increased. Users are automatically notified when an event that affects them occurs.

Portability: The ease with which a system or component can be transferred from one hardware or software environment to another. (IEEE, 1990)

Protocols: A common set of instructions that allows applications to communicate with one another. (IEEE, 1990)

Reliability: The ability of a system or component to perform its required functions under stated conditions for a specified period. (IEEE, 1990)

Reusability: The degree to which a software module, or another work product can be used in more than one computing program or software system. (IEEE, 1990)

Scalability: The ease with which a system or component can be modified to fit the problem area. (IEEE, 1990)

Three-tier architecture: The three-tier application model splits an application into three logical component types, presentation logic, business logic, and data access logic. In the 3-tier model client workstations execute a smaller part of the application logic talking to an application server (middleware). Given a request from the client, the application server talks to the database

server to complete the task. In this model, the application server now performs the largest part of the process logic for each client reducing network traffic since now most of the work is concentrated between the application server and the database server. (Gold-Bernstein, 1998)

Transaction processing (TP) monitor: Name for a class of products that provides a transaction execution environment on top of conventional operating systems.

Two-tier client/server: An application development approach that splits an application into two parts and divides the processing between a desktop workstation and a network -server.

Web browser: Software that interprets the HTML commands and displays the page contents to a client.

Wrapper: The enclosure that is used to wrap a legacy application to make the legacy application available as an implementation of ORB client applications.

References Cited

Anderson, M., (1998, 25 March). *The IEW marketplace: Key trends and electronic workplace architecture*, [Online], # pages. Available: GartnerGroup Interactive. <http://www.gartner.com> [May 28, 1999]

BEA Systems, Inc., (1998, August), *Speaking middleware: Your guide to enterprise middleware terms, techniques, and technologies*, BEA Enterprise Middleware Solutions Group.

Berson, Alex, (1996, April), *Client/server architecture 2nd edition*, J. Ranade Series on Communications, McGraw Hill.

Brown, Eric G., (1999, March), *Integrating business processes*, Forrester [Online], 27 pages. Available: Forrester. <http://www.forrester.com> [May 25, 1999]

Byrnes, Nanette, and Judge, Paul, (1999, June 28), "Internet anxiety", *Business Week*, page 78-88, McGraw-Hill.

Costa, Philip, (1999, January 6), *The web and distributed components*,

[Online], 9 pages. Available: Giga Information Group
<http://www.gigaweb.com> [May 15, 1999]

Costa, Philip, (1999, February 16), *Outlining the benefits of a layered application architecture*, IdeaBytes, [Online] 2 pages. Available: Giga Information Group, Available: <http://www.gigaweb.com> [June 1, 1999]

Dec, K., Mack, E., and Anderson, E., (1998, July 24), *From mainframes to distributed computing: The technical issues*, Strategic Analysis Report, Gartner Group, Inc., Spectrum for IT-Americas, document # R-05-4102.

Eckerson, Wayne, (1996), *Three-tier client/server architecture*, Patricia Seybold Group, [Online], 20 pages. Available:
<http://www.openvms.digital.com/whitepapers/3tier> [September 11, 1998]

Edwards, Jeri, (1999, February), *3-tier client/server at work*, John Wiley & Sons.

Enslow, B., and Schulte, R., (1998, August 19), *Outflank the competition by developing IT to build a 'zero-latency enterprise'*, Inside Gartner Group, [Online], 9 pages. Available: GartnerGroup Interactive
<http://www.gartner.com> [June 2, 1999]

Freidheim, Cyrus F., (1999, First Quarter), "The trillion dollar enterprise", 30 pages, *Strategy and Business*, Booz Allen and Hamilton, Reprint No. 99106.

Gilpin, Mike, (1998, December 30), *Distributed application platforms (thematic planning assumption)*, Enterprise Planning Assumptions Guide, Giga Information Group, document # T-1098-002.

Gilpin, Mike, (1999, March 24), *How to select an enterprise application integration solution*, Enterprise Planning Assumptions Guide, Giga Information Group, document # P-0399-008.

Gilpin, Mike, (1999, March 30), *Giga proposes ideal middleware architecture: Future direction for industry and customers*, Enterprise Planning Assumptions Guide, Giga Information Group, document # P-1298-009.

Gilpin, Mike, (1999, May 10), *Internet application integration: A new market to support cross-enterprise e-business*, Enterprise Planning Assumptions Guide, Giga Information Group, document # P-0599-005.

Gilpin, Mike, (1999, May 27), *Application servers are primary area of future*

middleware research interest for GigaWorld Europe attendees, IdeaBytes, [Online] 3 pages, Available: Giga Information Group <http://www.gigaweb.com> [June 1, 1999]

Gold-Bernstein, B., (1998, February), *Race to the middle*, Intelligent Enterprise Database Programming & Design [Online], 15 pages. Available: <http://www.dbpd.com> [March 22, 1999]

Hof, Robert, McWilliams, Gray and Saveri, Gabreille, (1998, June 22), "The click here economy", *Business Week*, page 20-29, McGraw-Hill.

Keene, Chris, (1998, October 21), *Putting muscle into middleware: Rapid development of high performance distributed systems using transactional application servers*, Waters Advanced Technology Conference presentation, [Online], 9 pages. Available: <http://www.watersinfo.com/events/atc> [June 29, 1999]

Kolesar, Peter, Van Rysin, Garrett and Culter, Wayne, (1998, Third Quarter), *Creating customer value through industrialized intimacy*, [Online], 27 pages. Available: <http://www.strategy-business.com/strategy> [March 9, 1999].

Magrassi, P., (1997, April 2), *Middleware: Dispelling the fog*, Technology Research Notes, [Online], 8 pages. Available: GartnerGroup Interactive. <http://www.gartner.com> [May 10, 1999]

Microsoft Corporation, Inc., (1997), *DCOM technical architecture*, [Online] 15 pages. Available: Microsoft, <http://www.microsoft.com> [May 2, 1999]

Open Management Group, (1999), *Introduction to CORBA*, [Online] 7 pages. Available: OMG <http://www.omg.com> [May 17, 1999]

Orfali, R., Harkey, D., & Edwards, J., (1997), *The essential client/server guide*, John Wiley & Sons

Papows, P., (1998), *Enterprise.com. Market leadership in the information age*, Perseus Books.

Pasternack, Bruce, and Viscio, Abert, (1998, Third Quarter), "The centerless corporation: A model for tomorrow", page 15-35. *Strategy and Business*, Booz Allen and Hamilton.

Schadler, T., Woodring, S., Overby, C., and Walker, J., (1998, June), *The Forrester Report: Getting apps to work together*, Forrester Research Volume

Nine, Number Three, [Online], 15 pages. Available: Forrester.
<http://www.forrester.com> [May 25, 1999]

Schulte, R., (1997, December), *Middleware: The glue that holds distributed computing together*, Enterprise Architecture Planning Group, [Online], 10 pages. Available: GartnerGroup Interactive. <http://www.gartner.com> [May 30, 1999]

Schulte, R., and Natis, Y., (1998, May 1), *There is no strategic OLTP middleware in 1998, Application Integration and Middleware Strategies*, [Online] 5 pages. Available: GartnerGroup Interactive. .
<http://www.gartner.com> [May 30, 1999]

Schulte, R., (1999, February) *Supergroups: IT suites for technology integration*, Applications Architecture/Frameworks, [Online], 6 pages. Available: GartnerGroup Interactive. <http://www.gartner.com> [May 30, 1999]

Special Report, (1998, November), "The global Internet 100 survey 1998", *Information Strategy*, [Online], 20 pages. Available: <http://www.info-strategy.com/G1100/index.html> [November 15, 1998]

Strauss, A. and Corbin, J., (1990), *Basics of qualitative research: Grounded theory procedures and techniques*, Sage.

Sweat, Jeff, (1999, April 26), "The integrated enterprise", *Information Week*, page 63-72.

The Institute of Electrical and Electronic Engineers, (1990, IEEE), *Standard computer dictionary: A compilation of IEEE standard computer glossaries*.

The Source for Java™ Technology, (1999), *Enterprise JavaBeans technology: Server component model for Java platform*, White paper, [Online] 25 pages. Available: <http://www.javasoft.com> [June 29, 1999]

The Writing Center at Colorado State University, (1999), *Introduction to content analysis*, [Online], 8 pages. Available: <http://www.colostate.edu/Depts/WritingCenter> [June 14, 1999]

Wall Street Middleware Working Group, (1998, April 14), *Successful inaugural: About the Wall Street middleware working group*, event press release, [Online], 3 pages. Available: <http://www.wsmwg.com/aboutwsmwg> [June 27, 1999]

Waters Enterprise Technology Group, (1999), *A look back at ATC' 98 a Waters Magazines executive event: Wall Street beyond Y2K*, [Online], 4 pages. Available: <http://www.watersinfo.com> [June 26, 1999]

Weil, Peter, and Broadbent, Marianne, (1998), *Leveraging the new infrastructure: How market leaders capitalize on information technology*, Harvard Business School Press.

Yin, R. K., (1989), *Case study research: Design and methods*, Sage.