TOWSON UNIVERSITY
COLLEGE OF GRADUATE EDUCATION AND RESEARCH

A SURVEY OF MIDDLEWARES

BY

TONI A. BISHOP

A THESIS PRESENTED TO THE FACULTY OF

TOWSON UNIVERSITY

IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

IN COMPUTER SCIENCE

AUGUST 2002

TOWSON UNIVERSITY
TOWSON, MARYLAND 21252

TOWSON UNIVERSITY
COLLEGE OF GRADUATE EDUCATION AND RESEARCH

THESIS APPROVAL PAGE

This is to certify that the thesis prepared by Toni A. Bishop entitled "A Survey of Middlewares" has been approved by her committee as satisfactory completions of the thesis requirement for the degree Master of Science in Computer Science.

_____	_____
Chair, Thesis Committee Dr. Ramesh K. Karne	Date
_____	_____
Committee Member Dr. Joyce Currie Little	Date
_____	_____
Committee Member Dr. Alexander L. Wijesinha	Date
_____	_____
Dean, College of Graduate Education and Research Dr. Jin Gong	Date

ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Ramesh Karne, my thesis committee chair, for his support and commitment to this project. His thoughts and assistances on the research phase of this thesis were invaluable. He provided many suggestions for topic and indicated sources where the information could be found. I would like to thank Dr. Joyce Currie Little, a member of the Thesis Committee, for her encouragement to continue on this topic even when the task seemed difficult and it took much longer than anticipated. I also want to thank Dr. Alexander Wijesinha for his supportive ideas throughout the thesis process.

I especially want to thank my husband, Richard Bishop, for his support during the long thesis process and for his editorial assistance.

ABSTRACT

A SURVEY OF MIDDLEWARES

TONI A. BISHOP

The term middleware has been used to describe many different type of software products. Middleware has been described as a helper software or "glue" that connects two or more softwares together. Since there is a wide variety of different softwares included in this big group called middleware, categories need to be assigned to further describe and delineate each of these types. The current literature has provided several different ways to categorize these packages. These categorizations seem to be conflicting and do not include all the types of middleware available today. Difficulties in categorizing middlewares are further exasperated by the fact that some middlewares can perform more than one service. We have produced a categorization based on the middleware considered as an application assistant or as an integration assistant. This categorization is further subdivided as to how each type operates or interacts with the supporting software.

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

KEYS TO SYMBOLS AND ABBREVIATIONS

ABR	Accessible Business Rules
ACID	Atomic, consistency-preserving, isolated and durable manner
ACL	Access Control List
ACM	Association for Computing Machinery
ADE	Application Development Environment
ADL	Architecture Definition Languages
ADO	Active Data Object (by Microsoft® Corporation)
ALE	Application Link Enabling
AMI	Asynchronous Method Invocation
ANSI	American National Standards Institution
APDU	Application Protocol Data Unit
API	Application Programming Interface
AS	Application Server
ASC	Application Scheduler Context
ASL	Architecture Specification Language
ASP	Application Specific Protocol
ATM	Asynchronous Transfer Mode
AT&T	American Telephone and Telegraph
BNA	Burrough's Network Architecture
BSD	Berkeley Socket
B2B	Business to Business
CAD	Computer-Aided Design
CASE	Computer-Aided Software Engineering
CAT	Crisis Action Team
CCM	CORBA® Component Model
CGI	Common Gateway Interface
CICS	Customer Information Control System
COM	Component Object Model (by Microsoft® Corporation)
CORBA®	Common Object Request Broker Architecture – specification from the OMG™ for this OO networked application communication with one another.
COS	Common Object Services
COTS	Commercially-Of-The-Shelf
CRM	Customer Relationship Management
CSCW	Computer Supported Cooperative Works
C2	Command and Control
C3P	Common Component-Connector Part

DACNOS	Distributed Academic Computing Networking Operating System
DBMS	Database Management System
DBPL	Database Programming Language
DCA	Distributed Network Architecture
DCE	Distributed Computing Environment – Open Software Foundation's specs for development, use and maintenance of distributed applications.
DCOM	Distributed Common Object Model – Microsoft's specs for creating objects and allow other programs or objects in a distributed system to effect objects.
DDM	Data Distributed Middleware
DDP	Data Delivery Protocol
DECS	Domino Enterprise Connection Services
DHCP	Dynamic Host Configuration Protocol
DHTML	Dynamic HyperText Markup Language
DII	Dynamic Invocation Interface
DLL	Dynamic Link Library
DNA	Distributed Network Architecture
DNS	Domain Name Service – used by the Internet
DOC	Distributed Object Computing
DRE	Distributed Real-time and Embedded system
DSI	Dynamic Skeleton Interface
DTD	Document Type Definitions
DTP	Distributed Transaction Processing (Protocol)
EAI	Enterprise Application Integration
EB	e-Business
EC	e-Commerce
EDI	Electronic Data Interchange
EJB™	Enterprise JavaBeans™
ERP	Enterprise Resource Planning
FAP	Format and Protocol
FIFO	First In, First Out
FT	Fault Tolerance
FTP	File Transfer Protocol
GIOP	General Inter-ORB Protocol
GOPI	Generic Object Platform Infrastructure
GOSIP	Government Open Systems Interconnect Profile
GUI	Graphical User Interface
GUID	Globally Unique Identifiers
HLA	High Level Architecture
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocols

IBM®	International Business Machines
ICE	Integrated Conversion Environment
IDC	Internet Database Connector (from Microsoft® Corporation)
IDE	Integrated Development Environments
IDL	Interface Definition Language
IDS	Intrusion Detection System
IEEE	Institute of Electrical & Electronics Engineers
IFIP	International Federation form Information Processing
IIOP	Internet Inter-Orb Protocol
IIS	Internet Information Server (by Microsoft® Corporation)
iMASH	Interactive Mobile Application Support of Heterogeneous clients
IOR	Interoperable Object Reference
IP	Internet Protocol
IPC	Interprocessor Communication
ISO	International Organization for Standardization
ISP	Internet Service Provider
IT	Information Technology
IVR	Interactive Voice Recognition
JDBC™	Java™ Database Connectivity
JMS	Java™ Messaging Service
JVM	Java™ Virtual Machine
J2EE™	Java™ 2 Enterprise Edition
LAN	Local Area Network
MA	Mobile Agents
MACE	Middleware Architecture Committee for Education
MAGIC	Middleware And Grid Infrastructure Coordination
MCP	Main Control Protocol
MDT	Mobile Data Terminals
MFC	Microsoft Foundation Classes
MIL	Module Interconnection Language
MIX	Mediation of Information using XML
MMM	Middleware for Method Management
MOCHA	Middleware based On a Code sHipping Architecture
MOM	Message-oriented Middleware
MOP	Meta-Object Protocol
MSMQ	Microsoft Message Queuing (by Microsoft® Corporation)
MSS	Multimedia System Services
MTS	Microsoft Transaction Server (by Microsoft® Corporation)
MVT	Mobile Virtual Terminal

NAS	Network Application Support
NDS	Netware Directory Services (by Microsoft ® Corporation
NSF	National Science Foundation
OASIS	Open Architecture for Secure, Interworking Services
OCL	Object Constraint Language
ODBC	Open Database Connectivity
ODP	Open Distributed Processing
OHS	Open Hypermedia System
OLE	Object Linking and Embedding
OLTP	Online Transaction Processing
OMA	Object Management Architecture
OMG™	Object Management Group™
ONC	Open Network Computing
ONE	Open Network Environment
OO	Object-Oriented
OODB	Object-Oriented DataBases
ORB	Object Request Broker
OS	Operating System
OSF	Open Software Foundation (Group formed in 1980s by IBM®, HP, and DEC)
OSI	Open Systems Interconnection
OTM	Object Transaction Managers
OTS	Off-The-Shelf
PAC	Platform Adapter Components
PC	Personal Computer
PDES	Parallel Discrete Event Simulation Systems
PGM	Pragmatic General Multicast
PKI	Public Key Infrastructure
POA	Portable Object Adapter
PREMO	PResentation Environments for Multimedia Object
QoS	Quality of Service
RAD	Rapid Application Development
RBAC	Role-Based Access Control
RMI	Remote Method Invocation
RMTP	Reliable Multicast Transport Protocol
ROI	Remote Object Invocation
RPC	Remote Procedure Calls
RT	Real-Time
SAA	System Application Architecture
SCSI	Small Computer Systems Interface

SIGACT	ACM Special Interest Group on Algorithms and Computation Theory
SIGOPS	ACM Special Interest Group on Operating Systems
SQL	Structured Query Language
SML	Synchronous Method Invocations
SMTP	Simple Mail Transfer Protocol
SNA	System Network Architecture
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol by W3C®
SOMA	Secure and Open Mobile Agent
SQL	Structured Query Language
SSL	Secure Socket Layer
TANGO	Temporal Adaptive Next-Generation query Optimizer and processor
TCM	Transactional Component Middleware
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TMO	Time-triggered, Message-triggered Object
TMOSM	TMO Support Middleware
TP	Transaction Processing
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locators
UVE	User Virtual Environment
VM	Virtual Machine
VRM	Virtual Resource Management
VSM	Virtual Service Machine
VTM	Virtual Task Machines
WAN	Wide Area Network
WAP™	Wireless Application Protocol
WBEM	Web-Based Enterprise Management
WLL	Wireless Local Loop
WOSA	Windows Open Services Architecture
WWW	World Wide Web
W3C®	World Wide Web Consortium®
XHTML	blend of XML and HTML
XML	eXtensible Markup Language
4GL	Fourth Generation Language

1.0 INTRODUCTION

The word "Middleware" has been used in many forms. It is used to describe a wide variety of software products. This can cause a problem in understanding exactly what is a middleware product. In the literature, there are many different middleware definitions. They range from a software layer between the operating system (and/or the network) and application to a "glue" between two applications. It has been described as a helper software or an essential part of a distributed system. A list of the quotes collected for the descriptions of middleware software are included in the appendix of this thesis.

One possible definition for middleware is the software that assists an application to interact or communicate with other applications, networks, hardware, and/or operating systems. This software assists programmers by relieving them of complex connections needed in a distributed system. It provides tools for improving quality of service (QoS), network fault tolerance, security (both local and network), message passing, file services, directory services, etc. in a transparent way to the user. [1]

In chapter 2, the history of middlewares and how it has developed over the years will be discussed. It appears that middleware products (not always known by that name) have been around since the 1970s. This chapter will cover the changes that have been made in the middleware arena since the 1970s.

The middleware architecture will be discussed in chapter 3. This architecture is a way of categorizing the different types of middleware products. This chapter contains the

[1] Geihs, K., Middleware Challenges Ahead, *IEEE Computer, Vol. 34, Issue 6*, Jun 2001, pg 26.

categorization several authors have proposed and compare it to a proposed new categorization. To show how these middleware are different, this chapter will identify each middleware type; discuss the operations performed by these middleware types; and explain how each is used. For most of the middleware products, their strengths and weaknesses were explored.

Standardization efforts and the future direction of these middleware types will be discussed in chapter 4. It is very important that the standardization steps identified in this chapter continue. Users of middleware are demanding more standardization. This is because the more standardized the middleware applications are, the easier it is to utilize them in a system, thereby reducing the manpower required to incorporate them. The future directions of middleware found in the literature will be provided in this chapter.

The final chapter will discuss the conclusions of the categorization and use of middlewares as described in this thesis. This conclusion chapter will summarize the proposed items above and indicate how they were explored in this thesis.

2.0 HISTORY AND DEVELOPMENT OF MIDDLEWARE

The first middleware products were not known as middleware per se. They were considered as helper tools for communication applications. Some of the earliest needs for middleware products were to assist in communication between companies within the same industries such as airlines. These airlines would have had multi-node networks as early as the late 1960s. Then, during the 1970s, most of the hardware vendors provided a way of networking many computers together. This networked group of computers was known as distributed systems.[2]

When large companies started to work on creating distributed systems, they needed a middleware product to aid in performance, control, data integrity, and ease of use. This was a major expense in both time and money and only large computer companies could take on this task at first. These companies would not regret this expense; however, since it propelled them forward into the ever demanding distributed computer system's market.[3]

According to John Charles' article, the first true middleware products were created in the early 1980s. Sun Microsystems®, Inc. developed a product based on remote procedure call (RPC) protocol to use with their Open Network Computing™ (ONC) system. This middleware type program allows one program to request that another program (in another computer) perform a task without having to be troubled about the network particulars.

---

[2] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 20.

[3] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 21.

These RPC functions work at the lower levels of the application programming interface (API), however the remote invoking function or method still needed to be developed.[4]

The Ada programming language was also considered the pioneer of the middleware concept since it provided runtime libraries that allowed for portability of the software across a wide variety of hardware platforms and operating systems (OS). The advances in Ada programming language's engineering features such as reusability, portability, and encapsulation made for a better class of reliable middleware products.[5]

In addition, in the early 1980s, supplemental middleware elements were developed to perform file and directory services due to the advancements of the hardware and networking technologies. Middleware products were being used more and more as industry moved from the client/server systems to the multitiered-distributed systems.[6] Early examples of the distributed systems, where middleware products may have been used, include Athena© at MIT, ITC/Andrew at CMU, DACNOS at University of Karlsruhe and IBM European Network Centre Heidelberg. These were available as early as 1985. [7]

The start of the open system movement soon brought about a push toward a standardization that would promote competition and lower the prices of these middleware products. In December 1980, the first draft of the Open System Interconnection (OSI) was published, which was authorized by the International Organization for Standardization

---

[4] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg 19.

[5] Royce, W., Boehm, B., and Druffel, C., Employing UNAS Technology For Software Architecture Education at the University of Southern California, *11th Annual Washington Ada Symposium*, Jul 1994, pg 115.

[6] McFall, C., An Object Infrastructure for Internet Middleware, *IEEE Internet Computing*, Mar/Apr 1998, pg 46.

[7] Geihs, K., Middleware Challenges Ahead, *IEEE Computer, Vol. 34, Issue 6*, Jun 2001, pg 25.

(ISO). It took several years to ratify this standard, but it failed to make significant impact on the evolution of middleware.[8] [9]

Since the early 1990s, advancements were made in the standard middleware platforms to increase asynchronous communication. The major push in this area was for event handling. It would provide timely responses to alarms, fault conditions, group communication, and multimedia applications. [10]

In 1989, the Object Management Group™ (OMG™) was founded. This group was the first to develop middleware specifications called Object Request Brokers (ORB). Since then, the major focus has been on detailing specific function of the middleware products to include queries, transactions, security, etc. These specifications continued to improve so that in 1996, authorization and access controls were included in the ORBs. By 1998, real-time specifications were added to the ORBs.[11]

With the rapidly growing importance of enterprise resource planning (ERP) applications, the middleware products continue to develop. The ERP applications assist many businesses to efficiently integrate new systems with legacy systems. They can handle the differences that occurs with multiple types of hardware, operating systems,

---

[8] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 21.

[9] Geihs, K., Middleware Challenges Ahead, *IEEE Computer, Vol. 34, Issue 6*, Jun 2001, pg 28.

[10] Bacon, J. and Moody, K., Toward Open, Secure, Widely Distributed Services, *Communication of the ACM, Vol. 45, No. 6*, Jun 2002, pg 60.

[11] Ceruti, M. and Thuraisingham, B., Dependable Objects for Databases, Middleware and Methodologies: A Position Paper, *Proceedings of the 5th International Workshop on Object-Orient Real-Time Dependable Systems, IEEE*, Nov 1999.

databases, network platforms, and applications used by the various parts of their organization.[12]

Most businesses have two options when it comes to modernizing their Information Technology (IT) systems – (1) convert to all new hardware, computing languages, platforms, and architecture or (2) get a facelift by using middleware products to aid in the assimilation of legacy systems. The first approach can be very expensive in terms of both dollars and manpower. It requires more than just converting the code to the new language, but may require the entire staff (or a major portion) to relearn the operation with the new applications. The second approach is usually chosen by most businesses since this saves both time and money. Otherwise, they would have to justify the major expense of the conversion in the first choice while not gaining many benefits for their customers.[13]

Of the current middleware products used today, the following is a list of core services that may be provided. However, no one middleware product will contain all of these services, nor should it. When selecting a middleware product, the programmer will only look for the services required to fulfill his/her needs. Some services provided by middleware products are:

- Identifiers – is a string of characters connecting the real world items to the data. These identifiers can apply to people, groups of people, objects, printers, etc. The important features of identifiers are the policies assigned to these identifiers and

[12] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg 17.

[13] Chiang, C., A Distributed Object Computing Architecture for Leveraging Software Engineering Systems, *Proceeding of the 16th ACM SAC2001 Symposium on Applied Computing*, Mar 2001, pg 653.

relationship between identified objects.[14]

- Authentication – ensures the action requested is associated with a particular subject or object. One way to perform authentication is to use the User IDs/Passwords. This certification process is not very secure and can cause inflexibility in applications, [15] but an encryption schema or encryption key can handle some of the security needs. [16]
- Naming Services/Directories – The naming services like DNS (Domain Naming Service) list the readable names associated with the numbers (IP address and port number). Directory services like NDS (Netware Directory Services or Microsoft Active Directory™) also provide a way of looking up names, but goes further by providing a general facility for looking things up, just like a telephone directory.[17] Essentially the directories are databases that contain information about real world subjects. Important issues include the schema, addressing of data, access permissions, and ownership of data.[18]

---

[14] Graham, J. and Cepull, J., Early Adopters, An Internet 2 Middleware Project, *ACM Proceedings of the Conference on User Services: Building the Future*, Oct 2000, pg 87.

[15] Graham, J. and Cepull, J., Early Adopters, An Internet 2 Middleware Project, *ACM Proceedings of the Conference on User Services: Building the Future*, Oct 2000, pg 87.

[16] Fraternali, P., Tools and Approaches for Developing Data-Intensive Web Applications: A Survey, *ACM Computing Survey (CSUR), Vol. 31, Issue 3*, Sep 1999, pg 249.

[17] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 78.

[18] Graham, J. and Cepull, J., Early Adopters, An Internet 2 Middleware Project, *ACM Proceedings of the Conference on User Services: Building the Future*, Oct 2000, pg 87.

- Authorization and Access Control – provide permissions to read, write, or update part or all of the data and/or the schema.[19]
- Certificates and PKI – include security access for multicast sessions, ubiquitous computing (access from any site), sharing and collaboration information, consistent access across variety of architectures, message handling between components, and monitoring of transactions.[20]
- Transaction Monitors – supervise a transaction to ensure that the transaction is an all or nothing process. This is especially important when more than one database is modified (insert, delete, or change a record or make modification to the schema of the database).[21]
- Process and Thread Controls – manage the multiple threads or processes that are started in the client and must be carried out in the server or servers. This is very important when using synchronous communication such as in RPC.[22]
- User and Real-World Interfaces – allow humans to receive information, images, sensors, and other items in a readable format. [23]
- Dynamic Load Balancing and Scalabilities – are ways of managing the access to multiple servers using a priority policy so that each server has a similar load. This

---

[19] Graham, J. and Cepull, J., Early Adopters, An Internet 2 Middleware Project, *ACM Proceedings of the Conference on User Services: Building the Future*, Oct 2000, pg 87.

[20] Graham, J. and Cepull, J., Early Adopters, An Internet 2 Middleware Project, *ACM Proceedings of the Conference on User Services: Building the Future*, Oct 2000, pg 87.

[21] Webopedia Website: http://www.webopedia.com/TERM/T/TP_monitor.html, found 2/20/2002.

[22] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 26.

[23] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 84.

allows for timely service of many more requests. The newest middleware can even learn new priority policies so that the servers are better utilized. [24] [25] Advance load-balancing features that satisfy demanding optimization are server transparency, stateful replicas, diverse load monitoring granularity, decentralized load balancing, fault tolerant load balancing, extensible load balancing algorithms, and on-demand replica activation. [26]

- Connectivity – "allows elements of application to interoperate across network links. Its main purpose is to bridge differences in underlying network protocols, system architectures, operating systems, programming languages, databases and other application services." [27]
- Data Transformation, Exchange, and Integration – provide a way to reorder fields, translate code pages, validate data, filter data, modify sophisticated messages, reformat messages, and perform division and combination of messages. [28]
- Integration Broker Services – "include message queuing, message dictionary, message warehouse, message transformation, message routing, and systems

[24] Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 126.

[25] Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

[26] Othman, O. and Schmidt, D., Issues in the Design of Adaptive Middleware Load Balancing, *ACM SIGPLAN Notices, Vol. 36, Issue 8*, Aug 2001, pg 205.

[27] Serbedżija, N., Developing Middleware for Web-aware Systems: Lessons Learned, *Proceedings of the Australasian Computer Science Conference*, Feb 2000.

[28] Mobile Middleware for Wireless Data Devices, http://yahoo.bitpipe.com/data/detail?id=973011263_806&type=RES&x=186610175, found 2/14/2002.

management capability. Many also provide publish-and-subscribe service."[29]

- Firewall Assistance – examines the data to determine if security standards are met and then grants or denies access to the client. It can filter and compress data to reduce access times and reduce the need for bandwidth. [30]
- Location Transparency – the applications should not have to know network and application address. It should be possible to move the application to a machine with a different network address without recompilation.[31]
- Message/Data Integrity – prevents messages or data from being corrupted, lost or duplicated. [32] This can be accomplished by storing the message/data until a confirmation that it has been transmitted to the requesting/correct destination has been completed.
- Re-configurability – allows for the rapid fluctuation in operational conditions and supports system evolution as requirements are updated. This should cover scalability, security requirements, etc. [33]

---

[29] Mobile Middleware for Wireless Data Devices, http://yahoo.bitpipe.com/data/detail?id=973011263_806&type=RES&x=186610175, found 2/14/2002.

[30] Ruggaber, R., Seitz, J., and Knapp, M., Π^2 - a Generic Proxy Platform for Wireless Access and Mobility in CORBA, *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*, Jul 2000, pg 193.

[31] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 23-24.

[32] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 24.

[33] Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr 2000, pg 173.

- Quality of Services (QoS) – provides properties such as "timeliness, precision, dependability, minimal footprint, and power consumption". [34]
- Context-aware computing – senses changes in execution context or environment and then selects the appropriate protocols according to specific policies.[35]

In addition to the above list, one author includes discovery protocols, resource-access protocols, public key management, event notification/logging, and role-based security policy administrator. [36]

[34] Wang, N., Kircher, M., and Schmidt,D., Applying Reflective Middleware Techniques to Optimize a QoS-enabled CORBA Component Model Implementation, *Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC)*, Oct 2000, pg 2.

[35] Tripathi, A., Challenges Designing Next-Generation Middleware Systems, *Communications of the ACM, Vol. 45, No. 6*, Jun 2002, pg 40.

[36] Tripathi, A., Challenges Designing Next-Generation Middleware Systems, *Communications of the ACM, Vol. 45, No. 6*, Jun 2002, pg 40.

3.0 TAXONOMIES OF MIDDLEWARE

Taxonomy is defined as the general principle of scientific classification. [37] There are many ways to classify middlewares. One of the problems of classifying is that middlewares are not just one service, but are a combination of several. Another problem is that there are a large variety of middlewares themselves and many different technologies behind them.

There have been several discussions in the literature about middleware classification. In his book, Chris Britton suggests communication as a way to classify middlewares. Later, he suggests a further classification of middleware by protocols (session versus sessionless). He then states "Clearly classifying by protocol is a complex undertaking, so perhaps looking at the programming interface would be a simpler and better approach." He finally finishes the classifying topic with this quote. "There are many ways to classifying middleware, none of which is entirely satisfactory."[38]

Another author (David Linthicum) also stated that dividing middlewares into categories is difficult. He suggests that the classifier needs to first understand the products and then try to fit them into categories. The classifier should also "identify the categories of

---

[37] Merriam-Webster, *Webster's Ninth New Collegiate Dictionary*, Springfield, Massachusetts, U.S.A., 1987, pg 1209.

[38] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 78 - 84.

middleware available and what purpose each category serves." His categories included database-oriented, virtual system, middle-tier, gateways, and Web-enabled. [39]

A third author (Jess Thompson) made his classification slightly different. Some of his separations had to do with communication, but he added programming execution and number of participants. His classifications are [40]:

- Advance peer-to-peer, module-to-module communication with 1-1 participation and Blocking for program execution
- Database gateway, 1-1, blocking
- Database replication, 1-1, blocking
- Database optimized SQL, 1-1, blocking
- Remote procedure calls, 1-1, blocking or non-blocking
- Object request brokers, 1-1, blocking or non-blocking
- Direct messaging, 1-1, non-blocking
- Message queuing, many-many, non-blocking
- Publish/subscribe, many-many, non-blocking

A fourth source of classification comes from a web page that divides the middleware into ten categories. This website was set-up to sell middleware products. At the time of this research, there were 282 different middleware products in their inventory. The categories are [41]:

- Message-oriented (MOM)
- Desktop access
- Object
- Database administration tools
- e-Commerce
- Data warehousing
- Legacy connectivity
- Application integration
- Transaction processing
- Data access (DAM)

---

[39] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, http://www.dbmsmag.com/9709d14.html, found 3/29/2001, pg 1.

[40] Thompson, J., Avoiding a Middleware Muddle, *IEEE Software*, Nov/Dec 1997, pg 94.

[41] KnowledgeStorm, http://www.knowledgestorm.com/, headquartered in Atlanta, GA, found 2/20/2002.

While these classifications were good, this thesis will describe a different approach. Throughout the research of this thesis, many different types of middleware became evident. We propose a new classification based on the implementations and applications. A particular middleware can be in an implementation type and an application type, but the descriptions of each category will make this major division clear. Figure 1 displays the major breakdown of these middlewares.

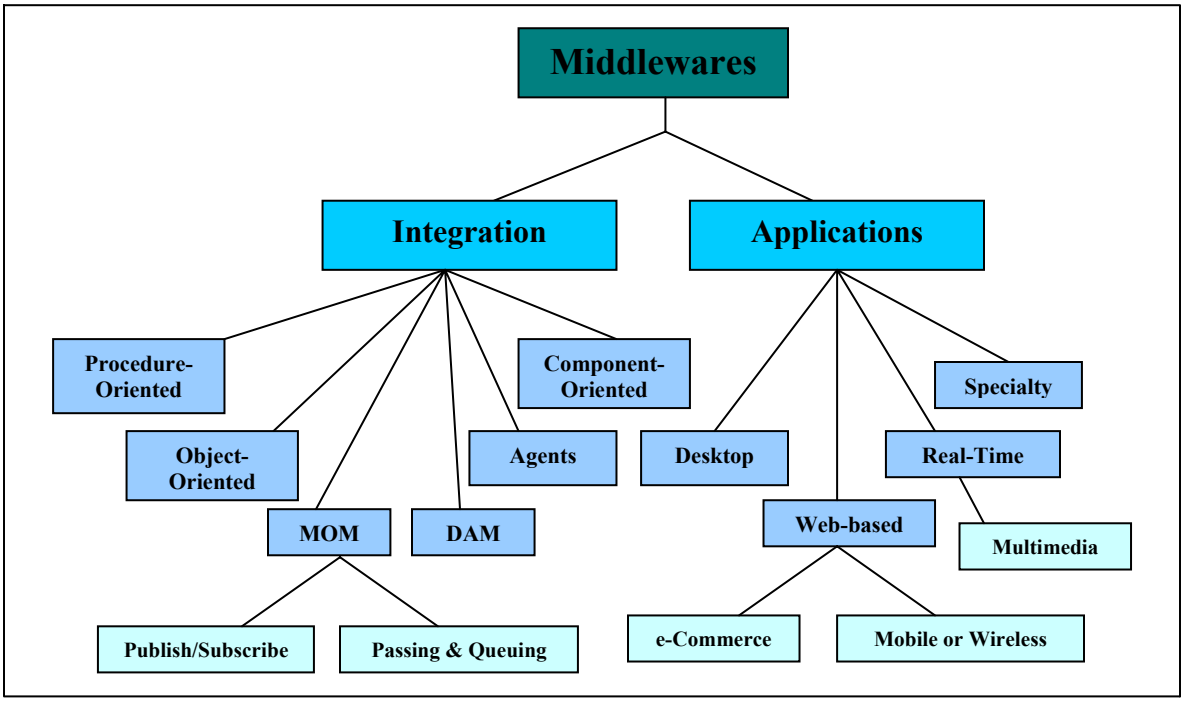

Figure 1 - Middleware Classifications

The implementation classification includes those middlewares that have a specific way of being executed. As referenced in the previous attempts at classification, each of these middlewares has different communication protocols. In the next section of this chapter, each of these middlewares (Procedure-oriented, Database-oriented, Message-oriented [MOM], Component-oriented, Agents, and Object-oriented) will be examined. The MOM has been divided into two subcategories – Message Passing/Message Queuing and

Publish/Subscribe. The examination of these implementation categories will include how these middlewares operate and how they are used.

The application classification includes middlewares that fit into specific type of application functions. These middlewares (Web, Real-time, Desktop, and Specialty) work specifically with an application. Two of these categories have subcategories. The Web has been subdivided into e-Commerce and Mobile/Wireless middlewares. These middlewares assist the Web application, but have additional services that need to be explored. The Real-time middleware has an additional subdivision of multimedia. Later sections of this chapter will describe each of these middleware and their implementation as associated with their given application.

3.1 PROCEDURE-ORIENTED MIDDLEWARE

Procedure-oriented middleware is characterized by a client converting the parameters of a procedure into a message, sending this message to the server (or host), who then converts the message back into the parameters. This converting of the parameters into a message is called marshalling. The stubs in the client send packets to the server that include the procedure call and its parameters. The server skeletons will convert the packets into the procedure calls (known as un-marshalling) and their parameters. The server processes the requested procedure and then places the results back into a message, which is sent back to the client. Additionally, any errors or exceptions are sent with these results. [42] [43] Figure 2 shows this process, which is identified as remote procedure call (RPC)[44].

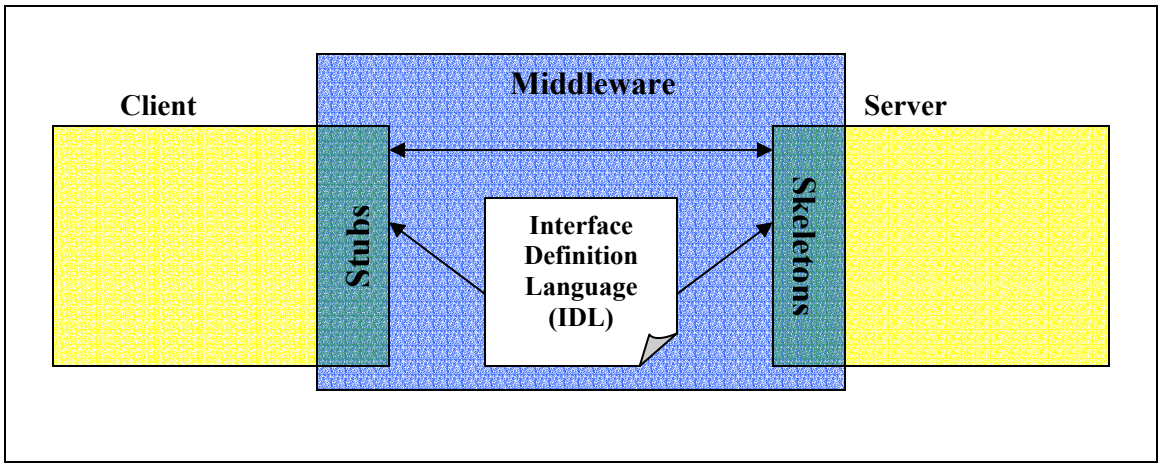

Figure 2 - Procedure-Oriented Middleware

---

[42] Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 124.

[43] Bernstein, P., Middleware*, Communications of the ACM*, *Vol. 39, Issue 2*, Feb 1996, pg 88.

[44] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 25.

The client spawns a thread or threads that enables the client to track the message. The client also suspends its operations until it receives a reply from the server. The Interface Definition Language (IDL), which is similar to a "header file" used in C programming, generates the client stubs and server skeletons to aid in the procedure call. Once the reply is accepted and there are no errors, the thread(s) are suspended and the procedure call is complete. [45] This type of communication can be analogous to a telephone conversation. Each side gets a turn to talk while the other side waits. [46]

Good examples of this type of middleware are Open Network Computing (ONC™) from Sun Microsystems®, Inc. and Distributed Computing Environment (DCE) from Open Software Foundation (OSF). [47]

The strengths of this procedural-oriented middleware are:

- It uses standard types of naming service, remote process, and returns a response.[48]
- It supports exceptions by returning a message that a failure occurred.[49]
- "It hides the intricacies of the network by using the ordinary procedure call mechanism familiar to every programmer." [50]

---

[45] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 25.

[46] Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

[47] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 25.

[48] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, http://www.dbmsmag.com/9709d14.html, found 3/29/2001, pg 2.

[49] Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 125.

[50] Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

- It can handle different types of data formats [51] and heterogeneous system-level services. [52]

These strengths are good, but are outweighed by the weaknesses. These weaknesses include:

- It is not scaleable because RPC does not have a replication mechanism.[53]
- It cannot return another program, which means it is not reflexive.[54]
- It is a rigid process because it is tightly coupled to the procedure. The client must wait to receive a response before continuing the process (known as synchronous communication). [55]
- It assumes that the network is available and the bandwidth is not a constraint. There is no optimization for the exchanges in messages. [56]
- It requires multi-threading. The client must send out a thread for the RPC. Likewise, the server also handles a thread for each client. This can cause lock-ups, especially with shared resources and a high volume of requesting clients.[57]

---

[51] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 26.

[52] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, http://www.dbmsmag.com/9709d14.html, found 3/29/2001, pg 1.

[53] Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 126.

[54] Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 124.

[55] Geihs, K., *Middleware Challenges Ahead*, *IEEE Computer, Vol. 34, Issue 6*, Jun 2001, pg 25.

[56] Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

[57] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 26.

- If the network goes down, the server is slow, or the message is lost, the client is left waiting unless time-outs are programmed.[58] The problem with time-outs is that the message may be sent more than once and this can create additional problems.

Some vendors are making changes to their procedural-oriented middlewares to correct some of these weaknesses. The major problem is thread management; it could be corrected by removing the use of threads. This would allow for asynchronous and non-blocking communication. A change like this would eliminate the last four weaknesses.

[58] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 26 – 27.

3.2 DATABASE-ORIENTED OR DATA ACCESS MIDDLEWARE

The characteristic of this middleware is the interaction of the application with local and/or remote databases (legacy, relational and non-relational), data warehouses, or other data source files. (This thesis will use the word database, but it will be referring all of the data storage types). This category of middleware includes transaction processing (TP) monitor, database gateways, and distributed transaction-processing middlewares.

The databases must maintain special requirements such as security (authentication, confidentiality, and access control), protection, and ACID properties. The ACID properties are Atomic (the transaction is an all or nothing process), Consistent (maintaining database schema and valid data constraints), Isolation (one transaction is unaware of another), and Durable (the transaction is completed and none of the updates are performed in the future). One way to ensure that these properties are maintained is to perform the "two-phase commitment".[59]

The image displayed in Figure 3 shows how the database middleware communicates with application and database(s). The application accesses the middleware using some type of application programming language (APL) or special SQL (Structured Query Language) commands. This middleware ensures the atomic property is maintained by handling the administration of each request. It divides and directs the data addition, modification, and/or deletion transactions to the correct database management systems (DBMS). This database middleware can even perform these requested transactions themselves if the

[59] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 30 – 31.

DBMS is unavailable or is unable to handle the transactions. Since the data can be maintained on more than one database, the middleware (specifically the transaction processing monitor) tracks the progress of each transaction and can request rollbacks when one part of the request fails. The middleware informs the requesting application of the status of the request and passes all returned data. Some middlewares even modify the appearance of the returned data in a format that makes the data easier to use by the application or the user.[60]

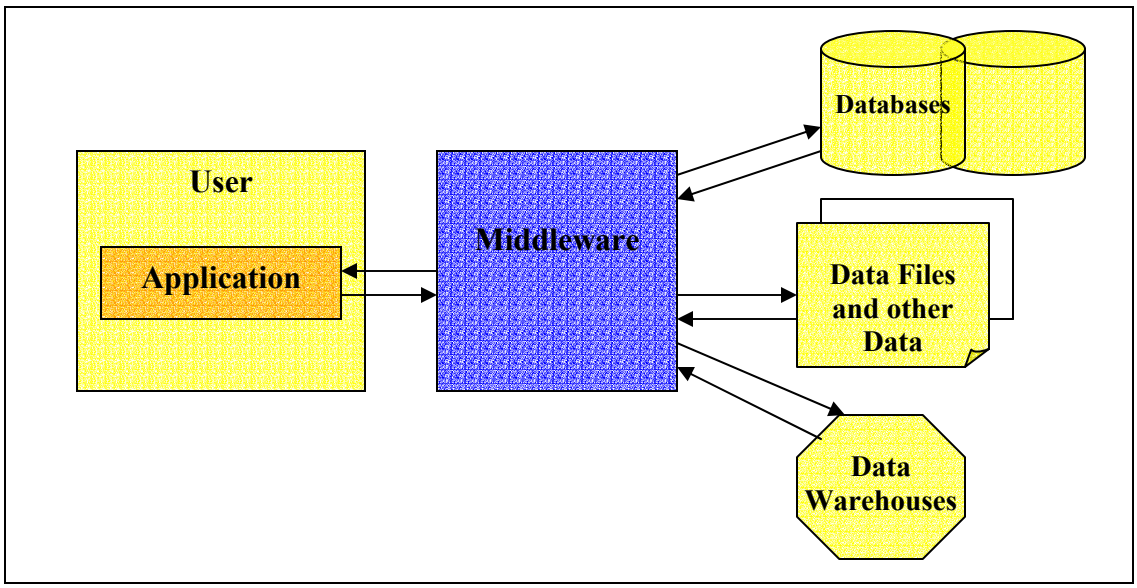

Figure 3 – Database-Oriented or Data Access Middleware

The middleware on the distributed network provides a way of traversing the operating system and network layers so that the communication is virtually transparent.[61] This means that the user may be on a Windows© OS and one database is on Unix® and another is on a Novell© network. The first middlewares of this type were created when one business

[60] Fraternali, P., Tools and Approaches for Developing Data-Intensive Web Applications: A Survey, *ACM Computing Survey (CSUR), Vol. 31, Issue 3*, Sep 1999, pg 249.

[61] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, http://www.dbmsmag.com/9709d14.html, found 3/29/2001, pg 3.

merged with another and their databases needed to be combined without converting the data or purchasing a new system.[62]

Some examples of this type of middleware are Oracle Glue® by Oracle® Corporation and OLE-DB by Microsoft® Corporation. The Open Database Connectivity (ODBC) and Java™ Database Connectivity (JDBC™) are database-oriented middleware standards and not middlewares themselves. [63] [64] Popular TP monitors include Tuxedo™ from BEA System Inc., Encina™ from Transarc™ Inc. (subsidiary of IBM® Corporation), and Transaction Server from Microsoft® Corporation. [65]

Strengths of the database-oriented middleware are:

- It allows communication between multiple sources and databases.
- It can convert the application programming language into something that is understandable to the target database(s).
- It is used to transparently provide a single interface across the network to databases on multiple types of platforms.
- It has the ability to perform queries on databases or communicate with the DBMS. It can also pass SQL directly to the database to invoke stored procedures and triggers.

---

[62] Linthicum, D., Database-Oriented Middleware, *DM Review*, Nov 1999, http://www.dmreview.com/editorial/dmreview/print_action.cmf?EdID=1560, found 3/29/2001, pg 3.

[63] Lewandowski, S., Framework for Component-Based Client\Server Computing, *ACM Computing Survey (SCUR), Vol. 30, Issue 1*, Mar 1998, pg 5.

[64] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 27.

[65] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, http://www.dbmsmag.com/9709d14.html, found 3/29/2001, pg 4.

- It can convert the response set into a format that is understandable to the requesting application.
- It can handle multiple and simultaneous requests.[66]

The weaknesses of some database middleware are:

- The more integrated middleware becomes, the more complex the software is for the programmer to incorporate it into the entire system.
- Some middleware have an "auto-commit" command included in their software. This can cause problems when more than one database is changed and a failure occurs. After the transaction has been committed, it is much harder to undo and put things back the way they were before the transaction. It can cause problems with atomic and isolation, two of the properties of ACID.[67]
- As with the procedural-oriented middleware, this communication is synchronous which means problems with thread management, applications waiting for responses, and network bandwidths.[68]
- Since the middleware must track all the requested transactions on multiple databases to ensure it is an all or nothing process, it will require a great deal of memory.[69]

---

[66] Linthicum, D., Database-Oriented Middleware, *DM Review*, Nov 1999, (http://www.dmreview.com/editorial/dmreview/print_action.cmf?EdID=1560), found 3/29/2001, pg 11.

[67] Linthicum, D., Database-Oriented Middleware, *DM Review*, Nov 1999, (http://www.dmreview.com/editorial/dmreview/print_action.cmf?EdID=1560), found 3/29/2001, pg 7.

[68] Thompson, J., Avoiding a Middleware Muddle, *IEEE Software*, Nov/Dec 1997, pg 93.

[69] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 32 – 34.

3.3 MESSAGE-ORIENTED MIDDLEWARE

Message-oriented middleware (MOM) is characterized by the message passing and message-queuing from one program to another. [70] Another type of MOM is considered publish/subscribe. MOM is different from procedural-oriented middleware in that it communicates asynchronously and there is no marshalling of the procedure parameters.

There are several differences between MOM and TCP/IP. These include names assigned to queues, queues are independent from programs, messages can wait for the network to become operational after a failure, the messages can be saved when the system fails, queues can manage resources, and queues can cooperate with a transaction manager (preventing messages sent when transaction fails).[71]

The popular MOMs are TIB/Rendezvous™ by TIBCO™, MessageQ [previously by Digital Equipment Corp (DEC) and now] by BEA™ Systems, MSMQ by Microsoft, MQSeries™ by IBM®, Allegris from Intersolv, Java™ Message Queue by Sun Microsystems®, Inc., and PIPES® from PeerLogic, Inc.[72] Additional MOMs include MessageQ™ by BEA™ Systems, JORAM by ObjectWeb Consortium, and Advance Queuing by Oracle® Corporation.[73]

---

[70] Campbell, A., Coulson, G., and Kounavis, M., Managing Complexity: Middleware Explained, *IT Pro*, Sep/Oct 1999, pg 27.

[71] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 34 – 35.

[72] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, http://www.dbmsmag.com/9709d14.html, found 3/29/2001, pg 4.

[73] Mobile Middleware for Wireless Data Devices, http://yahoo.bitpipe.com/data/detail?id=973011263_806&type=RES&x=186610175, found 2/14/2002.

Strengths of message-oriented middleware include:

- Its communication does not require large bandwidths, since it sends the same message (event) to all subscribers and isn't a point-to-point communication. [74]
- The information can be carried and acted upon by clients of heterogeneous distributed applications. [75]
- It provides asynchronous communication that is available even when one of the participants is not on line at the time of communication. This is performed using a "message store-and-forward capability." [76]
- The integrated message broker can have any of the following services: message queuing, message warehouse, message transformation, message dictionary, message routing, publish/subscribe, and some system management services. [77] The more services a MOM has, the better it is able to handle new and different requests.
- Persistent messages are able to recover in case of a failure. [78]

---

[74] Linthicum, D., Next Generation Middleware, *DBMS,* Sep 1997, http://www.dbmsmag.com/9709d14.html, found 3/29/2001, pg 4.

[75] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg. 18.

[76] Mobile Middleware for Wireless Data Devices, http://yahoo.bitpipe.com/data/detail?id=973011263_806&type=RES&x=186610175, found 2/14/2002.

[77] Mobile Middleware for Wireless Data Devices, http://yahoo.bitpipe.com/data/detail?id=973011263_806&type=RES&x=186610175, found 2/14/2002.

[78] Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

- Since the communication is asynchronous, there is no waiting for a response, so the client can do other things. In addition, there is no requirement for thread management. [79]
- This messaging system offers superior performance and scalability for multi-destination messages. This system can be used in financial market distribution, content distribution networks, and database replication.[80]

The weaknesses of the message-oriented middleware are:

- Persistent messages are slower than passing the message just through the memory.[81]
- This asynchronous communication is not good for time critical applications.[82]
- In the publish/subscribe MOM, the sender does not specifically address the receiver, but sends out a message to all subscribers. This may cause problems with security (especial in the area of confidentiality). [83]
- Some redundancy of transmission can cause problems with reliability in applications. [84]

---

[79] Uramoto, N. and Maruyama,H., InfoBus Repeater: A Secure and Distributed Publish/Subscribe Middleware, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

[80] Whetten, B., Message-Based Computing: The Fourth Wave of Integration, http://www.messageq.com/communications_middleware/whetten_1.html, found 2/14/2002.

[81] Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

[82] Whetten, B., Message-Based Computing: The Fourth Wave of Integration, http://www.messageq.com/communications_middleware/whetten_1.html, found 2/14/2002.

[83] Uramoto, N. and Maruyama,H., InfoBus Repeater: A Secure and Distributed Publish/Subscribe Middleware, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

[84] Whetten, B., Message-Based Computing: The Fourth Wave of Integration, http://www.messageq.com/communications_middleware/whetten_1.html, found 2/14/2002.

- Some of the MOMs are deployed using proprietary interfaces that are different from the standard Web interfaces, which can cause communication problems. [85]
- Messages sent don't always have a standard format therefore the subscriber may not understand the messages they receive.[86] There is a new standard called SOAP (Simple Object Access Protocol by W3C®), which helps with some of the standardization problems. [87]

[85] Whetten, B., Message-Based Computing: The Fourth Wave of Integration, http://www.messageq.com/communications_middleware/whetten_1.html, found 2/14/2002.

[86] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001, pg 35.

[87] Whetten, B., Message-Based Computing: The Fourth Wave of Integration, http://www.messageq.com/communications_middleware/whetten_1.html, found 2/14/2002.

3.3.1 MESSAGE PASSING/MESSAGE QUEUING MIDDLEWARE

In the message passing part of this MOM, the application sends out a message using the client MOM. This message may be intended for one or more other clients. There can be many clients sending requests to a server queue at any one time. The MOM server picks the requests off the queue (message broker) in some predetermined order. This order can be first-in/first-out (FIFO), based on a priority scheme, or determined in a load-balancing method. The MOM server has a filtering system that determines if it will process the message, send the message to another server, or destroy the message. Most MOM servers also provide persistent (store the message to disk) and non-persistent (store the message to memory) message queues. The MOM server acts as a router to the message and does not usually interact with it. The server MOM may provide directory services to allow the clients to look up other application and/or allow two clients to communicate as peers. (See Figure 4 for a diagram of this communication[88]).

The message queuing MOM looks on the MOM server to see if it has any messages waiting. If a message is available, it is retrieved from the queue and the MOM server removes any copies from its system. This is analogous to using mail or faxes. Messages can be held until requested, just like mail sits in a mailbox until the recipient removes it (or faxes sits in a pile on a machine until they are collected). The message can be sent even

[88] Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

when the recipient is not available. [89] This is much different from the previously reviewed middlewares, which used synchronous communication.

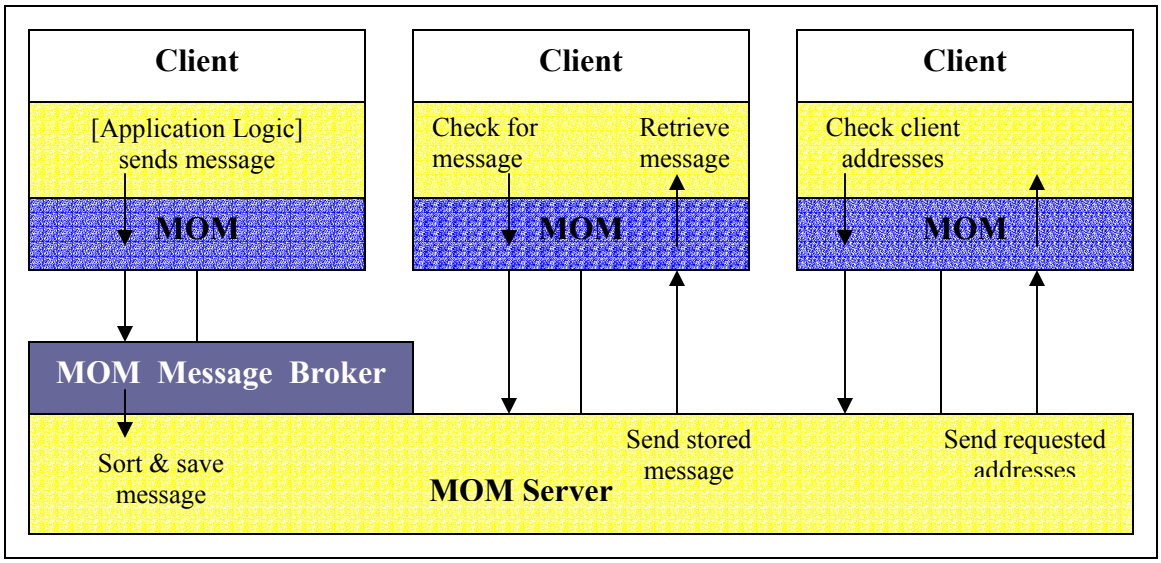

Figure 4 – Message Passing/Message Queuing Middleware

[89] Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

3.3.2 Publish/Subscribe Message-Oriented Middleware

The publish/subscribe MOM works slightly differently. This MOM is an event-driven process. If a client wants to participate, it first joins an information bus. Then depending on its function as the publisher, subscriber, or both, it registers an event listener in the bus. The publisher sends a notice of an event to the bus (on the MOM server). The MOM server then sends out an announcement to the registered subscriber(s) that data is available. When the subscriber requests from a specific publisher some data, the request is wrapped in a message and sent to the bus. The bus then sends an event to the publisher requesting the data.[90] This type of communication is displayed in Figure 5.

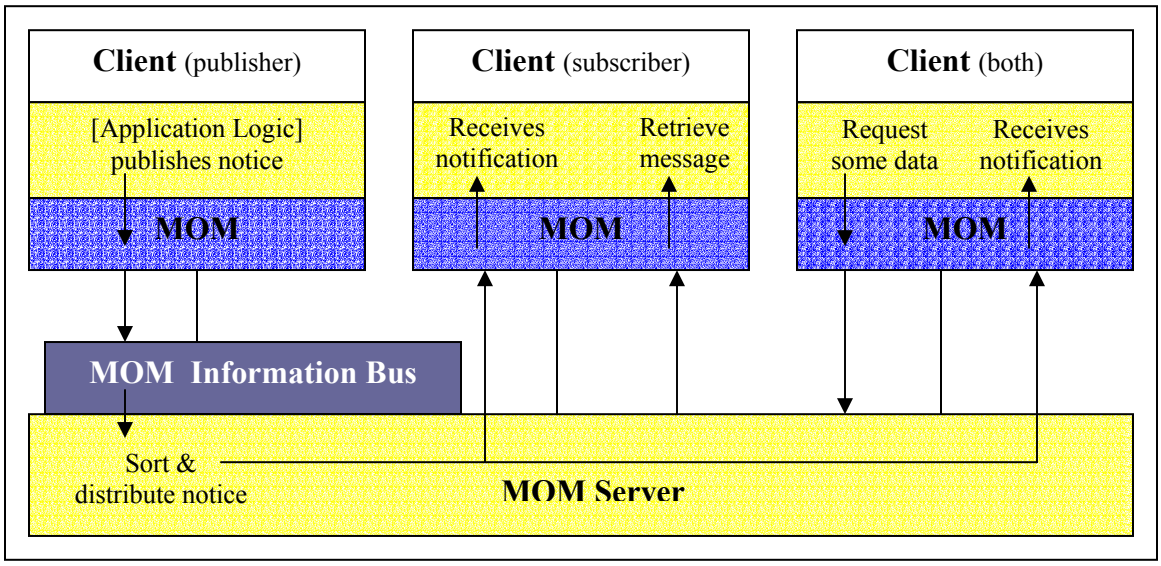

Figure 5 - Publish/Subscribe Message-Oriented Middleware

[90] Uramoto, N. and Maruyama,H., InfoBus Repeater: A Secure and Distributed Publish/Subscribe Middleware, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

3.4 COMPONENT-BASED (REFLECTIVE) MIDDLEWARE

A component is described as a "program that performs a specific function and is designed in such a way to easily operate with other components and applications."[91] This middleware is a configuration of components. These components are selected either at build-time or at run-time. The major contribution of this middleware is that it has an extensive component library and component factories, which support construction on multiple platforms. [92] Since the variation (i.e. resource availability, network connectivity, new required protocols) in the distributed computing environment is increasing yearly, software like reflective middleware will be needed to provide the ever increasing QoS requirements.[93]

Component-based middleware is built using basic constructs and standard elements of the Unified Modeling Language (UML™ by Rational® Software Corporation). [94] The UML™ has major elements that are divided into three categories (see figure 6):

- Components – computational units can be written in any programming language.
- Connectors – the interactive items between components (can be pipes or client/servers connections).

---

[91] Webopedia Website: http://www.webopedia.com/TERM/M/middleware.html, found 2/20/2002.

[92] Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr. 2000, pg 166.

[93] Kon, F., et al, The Case for Reflective Middleware, *Communications of the ACM, Vol. 45, no. 6*, Jun 2002, pg 33.

[94] Issarny, V., Kloukinas, C., and Zarras, A., Systematic Aid for Developing Middleware Architectures, *Communication of the ACM, Vol. 45, No. 6*, Jun 2002, pg 54.

- Configuration – application structure combines the components and connectors into a system.

The component-based middleware has formal specifications for many different component configurations, a way of selecting the configuration based on certain properties, and a set of rules to re-configure between different configurations. [95] The "reflective architectures provide expansion joints where new behaviors can be imposed." With these joints, changes to the operation can be made during run time. [96]

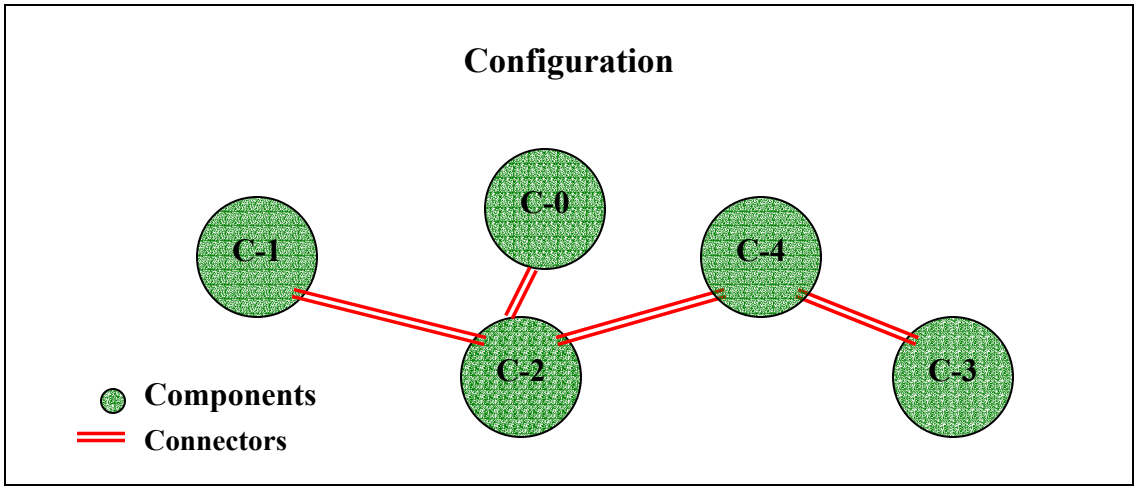

Figure 6 - Component-Based Middleware

The Common Component-Connector Part (C3P) is the kernel of this middleware platform; it is in charge of binding appropriate components and connectors to the application's generic references. These are then downloaded at run-time. The C3P has a Component Directory that identifies all components and connectors by their URLs. It also

[95] Fuentes, L. and Troya, J., Toward an Open Multimedia Service Framework, *ACM Computing Surveys, Vol. 32, No. 1*, Mar 2000.

[96] Thompson, C., et al., Intermediary Architecture: Interposing Middleware Object Services between Web Client and Server, *ACM Computing Surveys (CSUR)*, June 1999.

has an application directory that list the applications registered in the system. The client can use these directories to restart or join available services. [97]

Several types of this component-based middleware include Open CORBA® by OMG™, Dynamic TAO™ by http://www.cs.wustl.edu/~schmidt/TAO.html, [98], .NET® by Microsoft®, J2EE® by Sun Microsystems®, Inc. [99].

The strengths of the component-based middleware are:

- This middleware has more configurability and re-configurability. Configurability allows for the resolution of conflicting embedded systems, real-time systems, telecommunications, etc. Moreover, with the rapid changes in environmental conditions today, re-configurability is required for computing such as mobile computing, system evolution, and increase in loads.[100]
- It offers a lot of flexibility to meet the needs of a large number of applications.[101] This is accomplished because most of its tools are customizable, allow for integration of add-ins and allow for user-defined elements. [102]

---

[97] Fuentes, L. and Troya, J., Toward an Open Multimedia Service Framework, *ACM Computing Surveys, Vol. 32, No. 1*, Mar 2000.

[98] Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr. 2000, pg 164.

[99] Kon, F., et al, The Case for Reflective Middleware, *Communications of the ACM, Vol 45, no. 6*, Jun 2002, pg 33 & 35.

[100] Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr. 2000, pg 164.

[101] Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr. 2000, pg 166.

[102] Issarny, V., Kloukinas, C., and Zarras, A., Systematic Aid for Developing Middleware Architectures, *Communication of the ACM, Vol. 45, No. 6*, Jun 2002, pg 54.

- This middleware uses an object interface and can handle new protocols at run-time.[103]

The weaknesses of the component-based middleware are:

- It can offer too much flexibility so that it may compromise the integrity of the system by allowing too much access to the implementation details.[104] The use of multiple services can lead to complex interactions that can cause any of the following: "loss of information, non-terminations, causing deadlocks and livelocks, dangling resources, inconsistencies, and incorrect execution semantics."[105]
- There may be substantial development overhead for large and complex data structures because these structures must be identified in the interface definition language (IDL) of the middleware.[106]

---

[103] Campbell, A., Coulson, G., and Kounavis, M., Managing Complexity: Middleware Explained, *IT Pro*, Sep/Oct 1999, pg 26.

[104] Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr. 2000, pg 166.

[105] Venkatasubramanian, N., *Safe 'Composability' of Middleware Services*, Communications of the ACM, Vol. 45, No. 6, June 2002, page 50.

[106] Emmerich, W., Schwarz, W., and Finkelstein, A., Markup Meets Middleware, *7th IEEE Workshop on Future Trends in Distributed Computing Systems*, Dec 1999.

3.5 AGENTS

Agents are considered a middleware that consists of several components: entities (objects, threads), media (communication between one agent and another), and laws (rules on agent's communication coordination). Media can be monitors, channels, or more complex types (i.e. pipelines). Laws identify the interactive nature of the agents such as its synchronization or type of naming schemes. [107] An agent is capable of autonomous actions to meet its design objectives. This adaptability of the agent should be generic so that it covers a broad base of strategies. These strategies range from anytime algorithms, load balancing strategies, resource adaptability, and resource unaware applications. Developers should have access to these agents and the structured programming for adaptive systems.[108]

Agents perform tasks on behalf of the user. It can make decisions as to the best approach to accomplish the task. An example of this decision making process is to send a black-and-white picture as opposed to a colored picture, because the bandwidth is tight and the color in not needed for the particular application.[109] "The agent performs a name to object reference mapping and sends a handle back to the client. The client uses the handle to contact the server and receive the desired services." Several agent products are VisiBroker® by Inprise [now by Borland® Software Corporation] and Orbix-MT by Iona

---

[107] Ciancarini, P., Coordination Models and Languages as Software Integrators, *ACM Computing Surveys (CSUR), Vol. 28 Issue 2*, Jun 1996, pg 300.

[108] Ding, Y., et al, RAJA, *Proceedings of the 5th International Conference on Autonomous Agents*, May 2001, pg 332.

[109] Kleinrock, L., Breaking Loose, *Communications of the ACM*, Sep 2001, pg 45.

Technologies©.[110]

Dr. Wiederhold's OnTo-Agents establishes an agent infrastructure where each part adds information to the whole system. As shown in figure 7, several tools establish an exchange of information.[111] These include Ontology Construction tools (which provides all required terminology for information exchange), Webpage Annotation tools (which searches the web and finds appropriate terms of the ontology map for marking up web pages), and the Ontology Articulation toolkit (which coordinates information from other OnTo-Agent elements). The structure of this agent system is like a food chain that starts with the tools. The web pages from the Webpage Annotation tools are accessible by OnTo-Agents to accomplish its tasks. The Community Web Portal is built from the

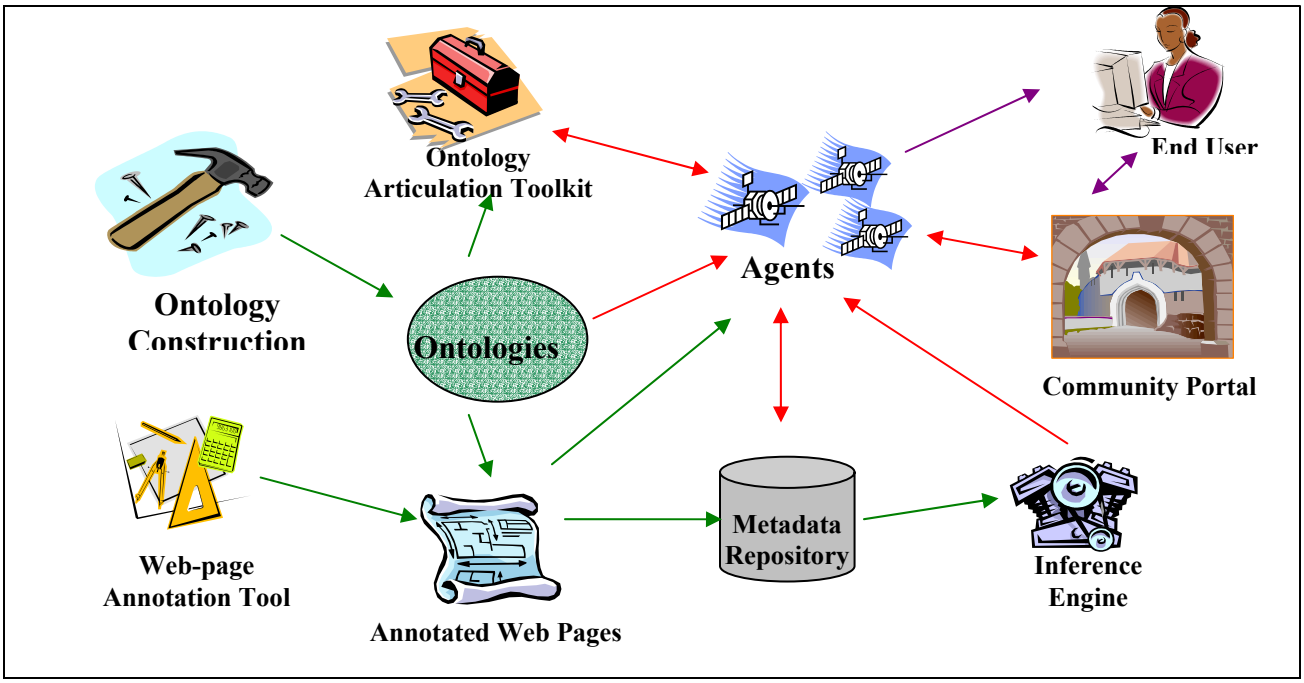

Figure 7 - OnTo-Agent System

[110] Petriu, D., et al, Using Analytic Models for Predicting Middleware Performance, *ACM Proceedings on the 2nd International Workshop on Software and Performance*, Sep 2000, pg 190.

[111] G. Wiederhold, et al., *OntoAgents – a Project in the DARPA DAML Program*, http://www-db.stanford.edu/Ontoagents, found 1/29/2002.

annotation process and is used to present information to the outside world in a succinct way. [112]

The OnTo-Agent needs additional components. One of these is the Inference Engine, which evaluates the rules, queries, and general assumptions. The End User places a request to the OnTo-Agent or queries the Community Portal to gain information immediately. The Metadata Repository stores information for the OnTo-Agents and web page information. [113]

The strengths of an agent middleware are:

- Agents can perform task on behalf of the user and therefore make decisions as to the best quality for the purpose and no better.
- Agents are adaptable so they can cover a broad range of strategies based on the computing environment around them.[114]

The weakness of an agent middleware is:

- Agents are complex, hard to understand, and will require a great deal of manpower to incorporate them into a system.

---

[112] G. Wiederhold, et al., *OntoAgents – a Project in the DARPA DAML Program*, http://www-db.stanford.edu/Ontoagents, found 1/29/2002.

[113] G. Wiederhold, et al., *OntoAgents – a Project in the DARPA DAML Program*, http://www-db.stanford.edu/Ontoagents, found 1/29/2002.

[114] Ding, Y., et al, RAJA, *Proceedings of the 5th International Conference on Autonomous Agents*, May 2001, pg 332.

3.6 OBJECT-ORIENTED MIDDLEWARE

Object and object-oriented (OO) middleware supports distributed object requests. The server object references client objects. The communication between these objects can be synchronous, deferred synchronous, and asynchronous using threading policies. Object middleware supports transactions by grouping requests from several client objects into a transaction. [115]

The OO middleware operates with the client object first making a logical method call to remote object. A local ORB proxy (also known as a stub) marshals the data and transmits it across to the server object. The server object receives the message and the remote proxy (known as the skeleton) unmarshals the data. The Broker acts as a middleman that contacts a number of data sources, obtains their reference IDs, collecting data, and sometimes reorganizing data. The data are submitted to a remote servant object where a particular process is performed. The return of results is marshaled by the skeleton, sent to the client object, and unmarshaled by the stub.[116] See Figure 8 below for the implementation of object-oriented middleware.

This appears to be similar to procedural-oriented middleware, but this middleware operates with objects. In addition, a Broker handles the data passed between the client and server objects.

---

[115] Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 124.

[116] Krishnamurthy, Y., et al, Integration of QoS-Enabled Distributed Object Computing Middleware for Developing Next-Generation Distributed Applications, *ACM SIGPLAN Notices, Vol. 36, Issue 8*, Aug 2001, pg 230.

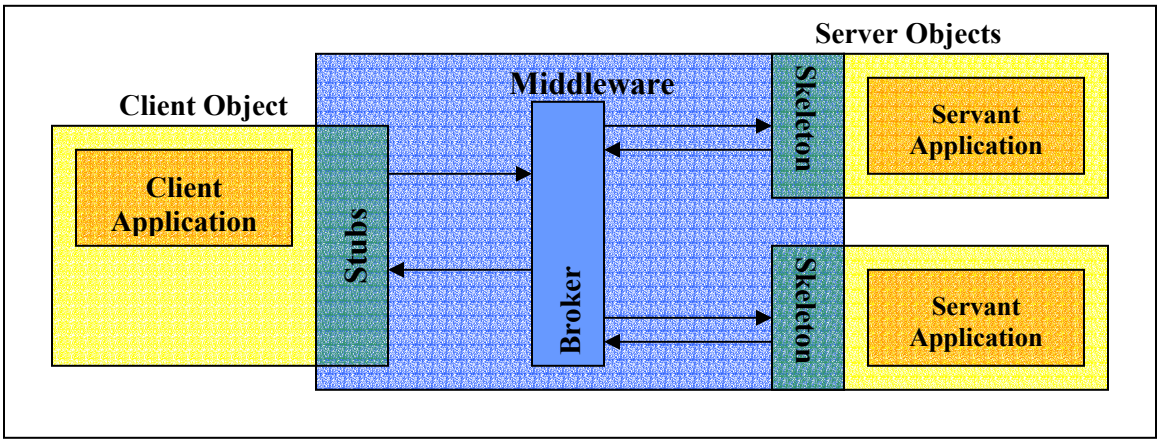

Figure 8 - Object-Oriented Middleware

Examples of OO middleware include CORBA® from OMG™ and COM/DCOM from Microsoft.[117] CORBA® has two main ways to locate another object in the system: Naming Services and Trader Service. The naming service is like the white pages of the phone book while the trader service operates more like the yellow pages. The trader services advertise certain characteristics. [118]

The strengths of object-oriented middleware are:

- It can access data stored in OO databases. These databases need services that are present in CORBA® (such as security and transaction management).[119]
- Object middleware supports load balancing and scaleable by determining which server object has the least load for the requested time. [120]

---

[117] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg. 19.

[118] Lewandowski, S., Framework for Component-Based Client\Server Computing, *ACM Computing Survey (SCUR), Vol. 30, Issue 1*, Mar 1998, pg 13.

[119] Caron, O., Carré, B., and Debrauwer, L., An Original View Mechanism for the CORBA Middleware, *IEEE Proceedings of the Technology of Object-Oriented Language and Systems (TOOLS 33)*, Jun 2000.

[120] Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 125.

- Object programming allows the storage of objects of various types, forms and states. Many lines of code would be needed to program for the storage of this variety of objects if OO was not employed. [121]
- "Self-managing distributed objects take the responsibility for their own resources, work across networks, and interact with other objects. ... Objects can generate events to notify other objects that an action should take place" such as synchronization objects that allow one thread to notify another thread of a process.[122]
- The Information Broker acts as middleman to collect data from multisource, organizes data collected, and provides many functions as a data source for other systems.[123]
- The cloaking system of object platforms (CORBA®, DCOM and RMI) allows programmers to know what is happening but makes it transparent to the operation.[124]
- Object middlewares communicate with one another. [125]

---

[121] Kuropka, D. and Weske, M., Transparent and Flexible Storage of Application Objects in CORBA Environments, *Addendum to the 2000 Proceedings of the Conference on Object-oriented Programming, Systems, Languages, and Applications*, Mar 2000, pg 169.

[122] Lewandowski, S., Framework for Component-Based Client\Server Computing, *ACM Computing Survey (SCUR), Vol. 30, Issue 1*, Mar 1998, pg 8 – 9.

[123] Altinel, M., et al, DBIS-Toolkit: Adaptive Middleware for Large Scale Data Delivery, *Proceedings of the 1999 International Conference on Management of Data (SIGMOD '99), Vol. 28, Issue 2*, May 1999, pg 544.

[124] Campbell, A., Coulson, G., and Kounavis, M., Managing Complexity: Middleware Explained, *IT Pro*, Sep/Oct 1999, pg 23.

[125] Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 124 – 125.

- OO middleware can support multiple operations at the same time using multi-threading.[126]

The weaknesses to object-oriented middleware are:

- "The object models underlying CORBA® and OLE do not contain all the necessary concepts for describing components and their assembly."[127]
- There is a requirement that the stub and skeleton are prelinked in the executables.[128]
- It may require the use of wrapper code to convert between the standard bus and some legacy interface.[129]

[126] Emmerich, W., Distributed Objects, *Proceedings of the 1999 International Conference on Software Engineering*, May 1999, pg. 666.

[127] Bronsard, F., et al, Toward Software Plug and Play, *ACM SIGSOFT Software Engineering Notes, Proceedings of the 1997 Symposium on Software Reusability, Vol. 22, Issue 3*, May 1997, pg 20.

[128] Campbell, A., Coulson, G., and Kounavis, M., Managing Complexity: Middleware Explained, *IT Pro*, Sep/Oct 1999, pg 23.

[129] Campbell, A., Coulson, G., and Kounavis, M., Managing Complexity: Middleware Explained, *IT Pro*, Sep/Oct 1999, pg 24.

3.7 WEB-BASED MIDDLEWARE

Web-based middleware that assists with browsing, uses interfaces that scout ahead to find pages of interest, and discerns user's changes of interest from browsing history. [130] It provides authentication service for a large number of applications[131] and interprocess communication that is independent from underlying OS, network protocols, and hardware platforms. [132] Some web-applications are tightly bound to this middleware. These middlewares are called application servers because they improve the "performance, availability, scalability, security, information retrieval, and support of collaborative administration and usage." Middleware may circumvent HTTP and connect directly to the application when this gains better communication between the server and client. [133]

Some core services provided by web-based middleware include directory services, electronic messaging (e-mail), billing, large-scale supply management, remote data access (to include downloads, program access, and browsing), and remote applications.[134] The Internet service allows users to connect to any device at any location transparently (just as

---

[130] MIT Media Lab: Software Agents: Projects, May 2001, http://agents.media.mit.edu/projects/, found 2/30/2002.

[131] KnowledgeStorm, http://www.knowledgestorm.com/, search word = middleware, headquartered in Atlanta, GA, found 2/20/2002.

[132] IT Pro, http://www.itprodownloads.com/filedownload?fileid=178578/, found 2/14/2002.

[133] Fraternali, P., Tools and Approaches for Developing Data-Intensive Web Applications: A Survey, *ACM Computing Survey (CSUR), Vol. 31, Issue 3*, Sep 1999, pg 248.

[134] Umar, A., et al, A Knowledge-based Decision Support Workbench fro Advanced E-commerce, IEEE, 2000.

one uses electricity without knowing where it comes from), but is available almost everywhere.[135]

According to one author, this middleware can be divided into two categories: client and server. The web-aware client middleware allows links to be developed for Java applets and ActiveX controls to connect to remote databases. Examples of the client middleware are XDB Systems Inc.'s JETConnect and Borland® Software Corporation's JBuilder™. [136] Thin clients on the web appear fatter by adding services such as caching, history mechanism, bookmarks, multiple views, SSL-based (Secure Socket Layer) security, and versioning.[137]

The server middleware products can connect the web server to a database server. This allows information to pass back to the client from the database and allows the mapping of database attributes to web pages for development of the application. Some server middleware products include WebDBC by StormCloud© Development Corp., IDC by Microsoft® Corporation, and ActiveWeb by Active Software Inc. [138]

A web-based middleware application must be able to handle many performance concerns. One of these is the fluctuation in the number of users as traffic increases. If the clients cannot get their information quickly, they tend to abort their request. The middleware needs to keep track of the state of the session between client and server and the

---

[135] Kleinrock, L., Breaking Loose, *Communications of the ACM*, Sep 2001, pg 45.

[136] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, (http://www.dbmsmag.com/9709d14.html), found 3/29/2001, pg 6.

[137] Thompson, C., et al, Intermediary Architecture: Interposing Middleware Object Services between Web Client and Server, *ACM Computing Surveys (CSUR)*, Jun 1999.

[138] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, (http://www.dbmsmag.com/9709d14.html), found 3/29/2001, pg 6.

client profile. This is important because the server can become overloaded when a huge number of concurrent clients are making requests.

Security is always an issue when using the web. The web itself is an insecure medium and there is a trust issue between clients and servers. Quality of service (QoS) is not always guaranteed over the web; therefore, some software (middleware in this case) must help to ensure quality in response time, availability, data accuracy, and consistency. And as always, there are legacy systems that need to be seamlessly integrated with newer computers.[139]

The web middleware may need to ignore the strict seven layers of the OSI's Internet architecture. Some applications may need interactions with nonadjacent layers; "context-aware applications may need the IP addresses or geographical location to make location-dependent decisions." Some applications need authentication information like encryption keys. Multimedia application may need to change the transport protocols. [140]

The web middleware can be subdivided into several categories. In this thesis, two major subcategories need to be discussed because of the particularities that exist. These subcategories are e-Commerce and Mobile Computing (also known as wireless web).

---

[139] Geihs, K., Middleware Challenges Ahead, IEEE Computer, Vol. 34, Issue 6, Jun 2001, pg. 26.

[140] Geihs, K., Middleware Challenges Ahead, IEEE Computer, Vol. 34, Issue 6, Jun 2001, pg. 29.

3.7.1 E-COMMERCE MIDDLEWARE

The term e-commerce pertains to the communication between two or more businesses or patrons and businesses performed over the web. This middleware controls access to customer profile information, allows the operation of business functions such as purchasing and selling items, and assists in the trade of financial information between applications. This business middleware can provide a modular platform to build the next generation of web applications. [141]

In 1999, The Yankee Group (a market research firm) reported that the e-commerce business had grown to a value of $138 billion. They estimated this would increase to $541 billion by the year 2003. The need for security, quality of service, cost-effective and speedy transactions, and transparency over diverse environments is essential. [142] These transactions should be conducted so that they eliminate paper trails, duplication on data processing, and re-keying of information.

The technology standard, XML (eXtensible Markup Language), is used to integrate, transform, and stream data over the Internet. HTML (HyperText Markup Language which is another web mark-up language) is used to define how elements will look, whereas, XML defines data element contents and identifies their structures (essentially data about data). The World Wide Web Consortium® (W3C®) in 1998 recommended that XML become a core standard in the communication of business transactions. XML describes documents containing "structured information such as e-Business transactions, relational databases

[141] KnowledgeStorm, http://www.knowledgestorm.com/, headquartered in Atlanta, GA, found 2/20/2002.

[142] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg. 17.

schemas, object metadata and APIs." XML is used for business-to-business (B2B) documents or web pages and will be a dominant force in business dealings for the beginning of the 21st century. [143]

The XML-based middleware helps integrate XML into the business solutions by assimilating existing or legacy systems that may not be XML-aware. In addition, it ensures the transformation of data between various formats including XML and creates a transparent, organized, and easy to use system. This allows companies to achieve the "full potential of XML data exchange and integration." This middleware can incorporate "a wide range of business applications including enterprise application integration (EAI), electronic data interchange (EDI), enterprise portals, and web services." [144]

The strengths of e-commerce middleware are:

- It enables the fast integration of various computing systems into a web-based business solution.
- It makes communication between businesses easier, cost-effective, and more secure.
- It allows customer service representatives to access data from multiple customer information systems. [145]

---

[143] DataMirror Whitepapers, Managing your Data the XML Way: Data Transformation, Exchange and Integration, http://www.datamirror.com/resourcecenter/download/dbxmlvision/pdfs/dbxmltransform/XML_solutions.pdf, found 2/14/2002.

[144] DataMirror Whitepapers, Managing your Data the XML Way: Data Transformation, Exchange and Integration, http://www.datamirror.com/resourcecenter/download/dbxmlvision/pdfs/dbxmltransform/XML_solutions.pdf, found 2/14/2002.

[145] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg. 18.

- It provides employees access to business information (e-mail, collaboration of customer data, meetings minutes, phone and office directories, sales trends, etc.) cheaply and efficiently.
- It offers customers a way to access a business (browse the company's products, compare prices between companies, make purchases, provide customer information, get purchase confirmation and delivery information, etc.).

The weaknesses of this middleware are:

- The current e-commerce middleware still has some web inherent problems such as security (authentication, certification, confidentiality, integrity and access controls), session drops, load balancing issues, timeliness of responses, availability, and accuracy.
- Even though this middleware is built to convert many data sources to allow for communication, no one middleware can cover all the different types of platforms, networks, data sources, or operating system.
- Some of these middleware products are expensive and small companies have to consider the cost factors (price of product, training, and maintenance) when (and if) they purchase these products.

3.7.2 MOBILE OR WIRELESS MIDDLEWARE

Mobile or wireless middleware integrates distributed applications and servers without permanently connecting (through wires) to the web. It provides mobile users secure, wireless access to e-mail, calendars, contact information, task lists, etc.[146] A mobile salesperson can check inventory and submit orders; a mobile repairman can access manuals, find and order repair parts, and check schedules; and mobile professionals can find hotel and restaurant accommodations, check appointments, and use contact lists while on the road. [147] Additional uses include national defense (troop movements), emergency and disaster management, remote control of appliances, as well as accessing the web. [148]

Some key issues of the mobile computing environment include bandwidth, reliability, delay, latency, error rate, user interface, processing power, interoperability, and cost. [149] The bandwidth and error rates increase exponentially in a wireless environment. The mobile system has other problems as well that include battery power fluctuations, network drops, moving out of signal range, location awareness, and the cost of communication. When the signal is transferred from one signal source to another, additional concerns arise like authentication (ensuring that this is the same user)[150] and checking for lost packets

---

[146] Fenestrae Mobile Data Server, http://yahoo.bitpipe.com/data/detail?id=1004968144_500&type=RES&x=1335401392 found 2/14/2002.

[147] Mobile Middleware for Wireless Data Devices, http://yahoo.bitpipe.com/data/detail?id=973011263_806&type=RES&x=186610175, found 2/14/2002.

[148] Varshney, U. and Vetter, R., Emerging Mobile and Wireless Networks, *Communications of the ACM, Vol. 43, Issue 6*, Jun 2000, pg 73.

[149] Kleinrock, L., Breaking Loose, *Communications of the ACM*, Sep 2001, pg 43 - 44.

[150] Geihs, K., Middleware Challenges Ahead, IEEE Computer, Vol. 34, Issue 6, Jun 2001, pg. 29.

(either sending or receiving). When using the TCP based communication infrastructure in a wired system, if a packet is not acknowledged within a set time frame, TCP would assume a network problem and slow down the transfer of packets. This is not necessarily true for the wireless communication. In a wireless system, packets can be lost due to hand-off from one signal tower to another, the high error rates, etc. Another problem with TCP is the setup and termination of the connection is expensive since it requires a three-way handshake. [151]

The mobile middleware would resolve these problems by first resending the lost packets and then continuing the communication at the previous rate. Some mobile middleware also filters and compresses the data to save on bandwidth. The location of the mobile user, the user's profile, and the current user's state should be available to the middleware as well.[152] According to one author, there are three major mobile services: user virtual environment (UVE), mobile virtual terminal (MVT), and virtual resource management (VRM). The UVE offers users a uniform view no matter the location or the transmitter of the signal. The MVT works to preserve the terminal execution state at different locations. Finally, the VRM automatically re-qualifies the bindings and accesses to resources and services at the new location as well as supporting load balancing and replication services. [153]

This research has found that the connection of mobile users can be either synchronous or asynchronous. The asynchronous connection appears to be the better and more popular

---

[151] Ruggaber, R., Seitz, J., and Knapp, M., *IT² - a Generic Proxy Platform for Wireless Access and Mobility in CORBA*, Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing, Jul. 2000, pg 192.

[152] Ruggaber, R., Seitz, J., and Knapp, M., *IT² - a Generic Proxy Platform for Wireless Access and Mobility in CORBA*, Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing, Jul. 2000, pg 193.

[153] Bellavista, P., Corradi, A., and Stefanelli, C., Mobile Agent Middleware for Mobile Computing, *Computer, Vol. 34, Issue 3*, Mar 2001, pg 73.

choice. The mobile user can submit a request, drop the connection (in numerous ways), and retrieve the results once reconnected. The reconnection can pose a security (reauthentication for both client and server) and state problems (reset the session state to previous) that needs to be handled.[154]

Mobile computing requires a very different set of protocols, tools, and procedures than a physically attached computer to the network. When the user moves from one place to another, the mobile middleware needs to terminate the connection from the old location, establish a connection at the new location, ensure the authentication of the user is the same at both locations, deal with heterogeneous systems, and check for and restore any lost information[155]. The mobile middleware should detect when the user plugs into or disconnects from a wired network so that it can best utilize the communication system. [156]

WAP™ (Wireless Application Protocol) adapts the existing Web pages for transmission over a wireless environment. The WAP Forum™, a consortium of wireless companies, has established the WAP™ specifications. The protocols are like other web protocols but use WML (Wireless Markup Language) to optimize the communication to mobile equipment. WAP Forum™ has defined six layers (as opposed to the seven of OSI):

- WAE –Wireless Application Environment (provides a browser that understands WML),
- WSP – Wireless Session Protocol (does HTTP functions and semantics),
- WTP – Wireless Transaction Protocol (supplies transaction services includes

---

[154] Geihs, K., Middleware Challenges Ahead, IEEE Computer, Vol. 34, Issue 6, Jun 2001, pg. 29.

[155] Bellavista, P., Corradi, A., and Stefanelli, C., Mobile Agent Middleware for Mobile Computing, *Computer, Vol. 34, Issue 3*, Mar 2001, pg 74.

[156] Engerman, G. and Kearney, L., Effective Use of Wireless Data Communications, *International Journal of Network Management, Vol. 8, Issue 1*, Feb 1998, pg 6.

delayed ACKs and concatenated PDUs),

- WTLS – Wireless Transport Layer Security (presents authentication and privacy),
- WDP – Wireless Datagram Protocol (gives a common interface for above layers by adapting to features of this wireless technology), and
- Wireless Networks (allows for transmission using wireless links). [157]

The WAP Forum™ has created a new developer registration and content verification program. A current list of the certified WAP™ clients and servers are located on http://www.opengroup.org/wap/cert/register.html. WAP™ specifications are standard for providing Internet communications and advanced telephony services. These services are on "digital mobile phones, pagers, personal digital assistants and other wireless terminals." Approved WAP™ 2.0 Specifications can be downloaded from http://www.wapforum.org/what/technical.htm. [158]

Examples of the mobile middleware are Mobitrix [159], LIME (Linda in a Mobile Environment) that is available from GNU Lesser General Public License[160] [161], and ExpressQ from Nettech© Systems, Inc. [162]

---

[157] Varshney, U. and Vetter, R., Emerging Mobile and Wireless Networks, *Communications of the ACM, Vol. 43, Issue 6*, Jun 2000, pg 79.

[158] WAP Forum, http://www.wapforum.org/, found 6/10/2002.

[159] Jain, R., Anjum, F., and Umar, A., A Comparison of Mobile Agent and Client-Server Paradigms for Information Retrieval Task in Virtual Enterprises, *IEEE*, 2000.

[160] Picco, G., Murphy, A., and Roman, G., Developing Mobile Computing Applications with LIME, *Proceedings of the 22nd International Conference on Software Engineering*, Jun 2000, pg 766.

[161] LIME Open Source License, http://lime.sourceforge.net, found 6/30/2002.

[162] Varshney, U. and Vetter, R., Emerging Mobile and Wireless Networks, *Communications of the ACM, Vol. 43, Issue 6*, Jun 2000, pg 78.

The strengths of mobile middleware are:

- It provides needed protocols, tools, and methods to handle the connection of a mobile device to the web. It takes care of changes in locations and sudden disconnects in service.
- It can provide services such as profile management (the communication device used and information services available), mailbox management (e-mail, fax, voice mail, etc.), cross-media translation (speech-to-text, text-to-speech, e-mail-to-fax, etc.) and others.[163]
- It (in the form of a proxy) may convert high-required bandwidth information into a lower bandwidth to save on transmission time and battery life. An example of this is the conversion of a picture initially in color to black-and-white.[164]

The weakness of wireless middleware is:

- In some developing countries, there is little or no means of collecting transmission from wireless devices. This may be rectified by using Wireless Local Loop (WLL). WLLs can provide several MHz of bandwidth for high-speed data transfer, Internet access, and basic telephone service. [165]

---

[163] Jain, R., Anjum, F., and Umar, A., A Comparison of Mobile Agent and Client-Server Paradigms for Information Retrieval Task in Virtual Enterprises, *IEEE*, 2000.

[164] Jones, C., et al, A Survey of Energy Efficient Network Protocols for Wireless Networks, *Wireless Networks, Vol. 7, Issue 4*, Sep 2001, pg 354 – 355.

[165] Varshney, U. and Vetter, R., Emerging Mobile and Wireless Networks, *Communications of the ACM, Vol. 43, Issue 6*, Jun 2000, pg 76.

3.8 REAL-TIME MIDDLEWARE

Real-time is characterized by the right data being provided on time otherwise it is no longer the correct data.[166] The real-time middleware supports time-sensitive request and scheduling policies. It does this with services that improve the effectiveness of the user applications. Real-time middleware can be divided into different applications using them. These include a real-time database application, sensor processing, and information passing. There are several types of real-time middleware products or services available today. Some of these (TANGO, OASIS, GOPI, DRE, and TMOSM) will be discussed in the rest of this section.

The real-time database services (like TANGO temporal middleware) include parsers, optimizers, translators, and execution engines. The parser can construct a query plan in algebraic functions. The optimizer (one of the most important services) determines the best approach to solving a problem to include identifying where the process resides, deciding how much time is required by several approaches, and determines the best approach for the query. The optimizer makes these decisions by collecting statistics on base relations and attributes (available indexes, minimum, maximum, medium, etc.). The translator takes the information from the optimizer, creates a query plan, and sends the plan to the execution engine. The execution engine either retrieves the information itself (can remove duplication and simplify queries), makes a request to the DBMS, or both. The execution

[166] Schmidt, D., Middleware for Real-Time and Embedded Systems, *Communications of the ACM, Vol. 45, No. 6*, Jun 2002, pg 43.

engine can then do more processing of the data before returning the final data to the requesting application.[167]

Another real-time service involves sensor data. The middleware captures information from multiple sensors and combines them into a single reliable reading. The middleware can then provide a view of any changes to the user in the format required and perform any required tasks. An example of this would be a burglar alarm system. It would alert a human operator of a suspected break-in at a remote location or it might sound an alarm to scare away the intruder. [168] [169]

Event-based middleware such as OASIS (Open Architecture for Secure, Interworking Services) is needed for many large-scale distributed systems. It can provide a timely response to alarm conditions and manage multimedia communications. This middleware may have event mediators (which presents the data in useful or desired format), event gateways (pathways for the data to be distributed), and event stores (a way of logging and auditing data from events). [170]

Real-time information passing middleware has increased dramatically with the introduction of the Internet, wireless networks, and new "dissemination-based applications". These applications are required to disseminate data to a large number of users like stocks and sports tickers, traffic information, electronic newspapers, and

---

[167] Slivinskas, G., Jensen, C., and Snodgrass, R., Adaptable Query Optimization and Evaluation in Temporal Middleware, *Proceedings of the 2001 ACM SIGMOND International Conference on Management of Data on Management of Data, May 2001, Vol. 30 Issue 2*, May 2001, pg 129.

[168] Srivastrava, M., Smart Kindergarten, *The 7th Annual International Conference on Mobile Computing and Networking 2001*, Jul 2001, pg 132.

[169] Stillerman, M., Marceau, C., and Stillman, M., Intrusion Detection for Distributed Applications, *Communications of the ACM, Vol. 42, Issue 7*, Jul 1999, pg 63.

[170] Bacon, J. and Moody, K., Toward Open, Secure, Widely Distributed Services, *Communication of the ACM, Vol. 45, No. 6*, Jun 2002, pg 60.

entertainment broadcasting. [171]

Another generic real-time middleware (GOPI – Generic Object Platform Infrastructure) has a set of modules: base (a foundation programming classes), thread (a collection of time sensitive, concurrent methods), msg (a "real-time interthread message-passing and buffer package"), comm (an architecture that considers ASP (application specific protocols), and bind (connection management and protocols for QoS). These modules are independent from each other and have separate libraries. One module can even replace another like the bind module could take over for the thread module in certain applications. [172]

Adaptive middleware is needed to resolve issues involving the Internet or mobile computing and time-sensitive application. It is very difficult to make the required time constraints when the web is congested or the communication has mobile problems (as discussed in section 3.7.2). The resource management of this type of middleware is both resource aware and allows for dynamic reallocation of resources. This middleware has to know the availability of the resources, what management policies are being enforced, and how the system is performing resource allocations.[173]

A challenge to the distributed real-time and embedded (DRE) system is to meet multiple quality of service (QoS) issues. The middleware must determine the appropriate level of each of the services given the circumstances such as environmental conditions within the system. Usually a time-critical application has the greatest emphasis on meeting the

---

[171] Altinel, M., et al, DBIS-Toolkit: Adaptive Middleware for Large Scale Data Delivery, *Proceedings of the 1999 International Conference on Management of Data (SIGMOD '99), Vol. 28, Issue 2,* May 1999, pg 544.

[172] Coulson, G., A Configerable Multimedia Middleware Platform, *IEEE Multimedia*, Jan/Mar 1999, pg. 63 – 64.

[173] Duran, H. and Blair, G., A Resource Management Framework for Adaptive Middleware, *Proceeding of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Mar 2000.

deadline for the data, but other services must be included to ensure that the mission needs are met. The middleware may trade off the level of one service to provide a greater level to another service. The task of this middleware is to determine the best mode of operation to provide a majority of the required services at the highest level given the computing environment The middleware can determine this by either statically (prioritizing services) or dynamically (optimizing to changes in the environment or requirements).[174]

The TMO (time-triggered message-triggered object) programming scheme has tools and techniques to develop real-time applications. It was established to eliminate conventional object's limitations in programming. TMO has a unifying approach to efficiently design and implement distributed applications with either real-time or non real-time requirements. A middleware has been developed from this scheme called TMOSM (time-triggered message-triggered object support middleware). This middleware can adapt to many different platforms and support all types of components into one concise and flexible system.[175] TMOSM has two types of threads: application and middleware, which activates when called for a specific time-slice. [176]

Strengths of time-dependent middleware are:

- They provide a decision process that determines the best approach for solving time-sensitive procedures.[177]

---

[174] Schmidt, D., Middleware for Real-Time and Embedded Systems, *Communications of the ACM, Vol. 45, No. 6*, Jun 2002, pg 44.

[175] Gimenez, G. and Kim, K., A Windows CE Implementation of a Middleware Architecture Supporting Time-Triggered Message-Triggered Objects, *Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC01)*, Jun 2000.

[176] Kim, K., et al, A CORBA Service Enabling Programmer-Friendly Object-Oriented Real-Time Distributed Computing, *IEEE 5th International Workshop on Object-Oriented Real-Time Dependable Systems*, Nov 1999.

- They can assist the operating system in the allocation of resources to aid time-sensitive applications to meet their deadlines. [178]
- TMOSM uses most operating system's kernel so COTS (commercial off-the-shelf) products' kernel can be utilized in the development of the distributed application.[179]

Weaknesses of the real-time middleware are:

- It is difficult to integrate security and fault tolerant services with real-time computing. This is because additional processing (sometimes extensive) needs to be performed to provide the security and fault tolerant services.[180]
- Because the middleware may need to manage threads that will affect the scheduler (inside the kernel), extra execution overhead will be incurred. [181]

The future trend is to have more and more time sensitive matter available to users. This is true due to the fact that business are more responsive to changes in the market, users want more up-to-the-minute information to make decisions, and more information is provided in a media that requires real-time applications. The multimedia area has become

---

[177] Slivinskas, G., Jensen, C., and Snodgrass, R., Adaptable Query Optimization and Evaluation in Temporal Middleware, *Proceedings of the 2001 ACM SIGMOND International Conference on Management of Data on Management of Data, May 2001, Vol. 30 Issue 2*, May 2001, pg 127.

[178] Duran, H. and Blair, G., A Resource Management Framework for Adaptive Middleware, *Proceeding of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Mar 2000

[179] Gimenez, G. and Kim, K., A Windows CE Implementation of a Middleware Architecture Supporting Time-Triggered Message-Triggered Objects, *Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC01)*, Jun 2000.

[180] Ceruti, M. and Thuraisingham, B., Dependable Objects for Databases, Middleware and Methodologies: A Position Paper, *Proceedings of the 5th International Workshop on Object-Orient Real-Time Dependable Systems, IEEE*, Nov 1999.

[181] Gimenez, G. and Kim, K., A Windows CE Implementation of a Middleware Architecture Supporting Time-Triggered Message-Triggered Objects, *Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC01)*, Jun 2000.

so important that middleware is now required to provide a real-time response. In this thesis, multimedia middleware has been designated as a subset of real-time middleware and is discussed in section 3.8.1.

3.8.1 MULTIMEDIA MIDDLEWARE

Multimedia middleware reliably handles a variety of data types. These types include speech, images (pictures, GPS outputs, etc.), natural language processors (translators and teleprompters), music, and video. The data need to be collected, integrated, and then delivered to the user in a time sensitive manner.[182] Multimedia systems can integrate a mixture of logical and physical devices. Physical devices may include video editors, cameras, speakers, and processing devices (data encoder/decoders or media synthesizers).[183] A real-time task may be described by its arrival time, deadline, worst-case execution time (without contention), and criticality (i.e. hard or soft time requirements).

Applications of this middleware type include: teleconferences, avionics mission computing [184], virtual environmental controls, network management, automated factory management, and military command and control systems. Each application has a critical time factor that must be met to allow operations to continue or a way to resolve missing or corrupt data. Real-time middlewares were created to meet these time critical factors by either bypassing the current systems or creating an environment that eliminates time wasting operations.

---

[182] Mills, K., Introduction to the Electronic Symposium on Computer-Supported Cooperative Work, *ACM Computing Surveys (CSUR), Vol. 31, Issue 2*, Jun 1999, pg 110.

[183] Duke, D. and Herman, I., A Standard for Multimedia Middleware, *ACM Multimedia'98*, Aug 1998, pg 384.

[184] O'Ryan, C., et al, The Design and Performance of a Pluggable Protocols Framework for Real-Time Distributed Object Computing Middleware, *IFIP/ACM International Conference on Distributed System Platforms*, Apr 2000, pg 373.

One important time-sensitive multimedia middleware assists in the distributed video-on-demand services. These services require a process to open new connections, ensure payment for services to the provider, and most importantly, ensure the quality of service (QoS) delivered to the customer. This is performed even though there may be problems with hardware or networks. [185] Aspects of QoS management involve mapping, negotiation and resource allocation at start-up, data transfer time, and error correction. [186]

GOPI (Generic Object Platform Infrastructure) middleware supports multimedia applications. It implements many configurable mechanisms to improve performance and reliability. Since audio and video streams require little or no marshalling, this middleware implementation will then incur very little penalty in performance. [187]

When multimedia is transferred over the Internet, TCP or UDP (User Datagram Protocol) are not good as communication protocols. Multimedia applications require a high throughput and strict delay and jitter specifications. Audio/video data have maximum acceptable delay times, loss-rates (error rates), and require very large bandwidths for the compression algorithms. There are inefficiencies when data are moved in or out of kernel space. Middleware may operate without the use of the OS kernel to prevent this ineffective process.[188]

Another approach is to use Open Hypermedia System (OHS) middleware. This middleware allows "an open set of structural abstractions to be served to the application to

---

[185] Astley, M., Sturman, D., et al, Customizable Middleware for Modular Distributed Software, *Communication of the ACM, Vol. 44, Issue 5*, May 2001, pg 99.

[186] Coulson, G., A Configerable Multimedia Middleware Platform, *IEEE Multimedia*, Jan/Mar 1999, pg. 62.

[187] Coulson, G., A Configerable Multimedia Middleware Platform, *IEEE Multimedia*, Jan/Mar 1999, pg. 75.

[188] Conrad, C. and Stiller, B., The Design of an Application Programming Interface for QoS-based Multimedia Middleware, *Proceeding of the 22nd IEEE Conference on Local Computer Networks (LCN '97)*, Nov 1997, pg 277.

the standard 'node, anchor, link' set of abstractions." This process can be very flexible when taking advantage of advanced functions available in OHS. This middleware can work with the Internet when the OHS integrations are also applied to the web applications.[189]

[189] Nürnberg, P. and Ashman, H., What was the question? Reconciling open hypermedia and World Wide Web Research, *Proceeding of the 10th ACM Conference on Hypertext and Hypermedia*, Feb 1999, pg 88.

3.9 DESKTOP MIDDLEWARE

This middleware makes variations in the presentation of the data as requested by a user. This is performed by tracking and assisting applications. The user accesses data through Microsoft® Windows Application from various applications and sources (such as flat files, MCP/AS-based data, etc.).

This middleware also manages any transport services (e.g. terminal emulation, files transfer, printing services). The desktop middleware provides backup protection and other operational background functions with minimal disruption. [190] It assists the operating system in providing needed services to increase performance.

Additional desktop middleware services include graphics management, sorting, character and string manipulation, records and files management, directory services, database information management, thread management, job scheduling, event notification services, software installation management, encryption services, and access control. This middleware also includes presentation, invocation, and repository services. The presentation service allows for the same look-and-feel of the GUI in different applications and integration of compound documents. [191]

Strengths of desktop middleware are that they give a similar look-and-feel to many different applications and provide multiple services to make computing easy to use and control.

---

[190] KnowledgeStorm, http://www.knowledgestorm.com/, headquartered in Atlanta, GA, found 2/20/2002.

[191] Bernstein, P., Middleware*, Communications of the ACM, Vol. 39, Issue 2*, Feb 1996, pg 91.

3.10 SPECIALTY MIDDLEWARE

Several types of middleware provide for specific needs. Since they may not fit any of the categories above, this category was created to hold these special types. Four of these stand out, and even though there appear to be more types on the web, this thesis will cover only these.

The first is called multi-campus system middleware. This middleware uses the policies and relationships around common identifiers, search defaults, and directory configurations. People (applicant, student, alumni, faculty/staff, and retiree) are assigned permanent and unique identifiers. This means that no matter where the person goes within the system the same information will follow them. The middleware works with the database transparently. It controls the security (authentication of user), access, and privacy (only allow certain users to see sensitive information) of the database. It also consists of advance tools like desktop video, distributed computing systems, collaboration environments. Essentially, it makes many computing activities much simpler. One way is by removing the need for multiple passwords and the annoyance and security problems associated with them. [192]

Another special middleware is called medical middleware. It covers enterprise issues such as security, directories, and authorization. It is a balance of institutional and medical enterprises. This middleware allows the sharing of data between institutions, medical centers, affiliated hospitals, state and Federal regulating and certifying bodies,

[192] Internet2, http://middleware.internet2.edu/multicampus/ found in 2/14/2002.

organizations, insurance companies, medical researchers, etc. This middleware covers issues such as standards, common operational process and policies, etc. It manages the gateways to other computing systems (Peer Institutions, Corporate Collaborators, etc.) and assists with the performance of the enterprise system. [193]

The third middleware is Ontology middleware. This middleware provides many services that allow users to easily employ and integrate ontological technologies into existing and future database systems. These services include accessing ontologies, ontology upgrading, query services, integration databases, etc. There are two major activities to be performed on an ontology system: development and management. The development activities performed are knowledge acquisition, editing, browsing, integration, merging, evaluation, implementation, etc. The management activities include configuration management, ontology evolution, ontology libraries, scheduling, and documentation, etc.[194]

The fourth specialty middleware discussed in this thesis is called PDES (Parallel Discrete Event Simulation Systems) middleware. This is used to simulate extensive applications (telecommunication networks, transport grids, battlefield scenarios, etc.). The middleware is used to enhance the visualization system with specific views, and to manage requirements needed with the simulation. It provides an accurate picture of the system being monitored by showing specific values.[195] These consist of the amount of time

---

[193] Internet2, http://www.internet2.edu/presentations/20020201-camp-blatecky.ppt found in 2/14/2002.

[194] Internet2, http://middleware.internet2.edu/internet2-and-sparc.html found in 2/14/2002.

[195] Carothers, C., et al, Visualizing Parallel Simulations in Network Computing Environments, *Proceedings of the Conference on Winter Simulation*, Dec 1997, pg 110 – 111.

needed to advance the simulation by a single unit of time; percent of rolled back events for late arrivals and processing anti-messages; percent of events aborted; and memory usage.[196]

[196] Carothers, C., et al, Visualizing Parallel Simulations in Network Computing Environments, *Proceedings of the Conference on Winter Simulation*, Dec 1997, pg 114.

4. MIDDLEWARE ISSUES

In this chapter, the standardization efforts in the middleware area will be discussed. Standardization is very important because understanding a middleware is easier when the middleware follows established standards. A list of metrics should be established so that middleware of the same kind can be compared and the applicable manpower needs can be determined.

Future trends of middlewares will also be discussed in this chapter. There will continue to be improvements with the current middlewares, but also a requirement for new middlewares so that programming of systems will become easier, more reliable, and provide for required services.

4.1 STANDARDIZATION OF MIDDLEWARES

The standardization of middlewares was first performed by ISO in 1980 by creating the first draft of the OSI. It took several years to ratify this standard but it had little effect for the standardization of middleware.[197] [198] The next attempt of standardization was in 1989 when the OMG™ was founded and they developed middleware specification known as ORBs. In 1996, the specifications were improved to include security measures to ORBs. Then in 1998, the ORBs included real-time services.[199]

With the increasing use of middlewares, there are still many challenges to this standardization issue.[200] It is crucial to determine the different types and functions of the current middlewares. Two organizations (X/Open® and OMG™) are working on standards for integration, naming conflicts resolution, and OS and communication services interfaces.[201]

One of the ways to standardize middlewares is to use components. Components work with a variety of systems and can supply dependable services. The important thing is to

---

[197] Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley, ISBN 0201709074, April 2001, pg 21.

[198] Geihs, K., Middleware Challenges Ahead, IEEE Computer, Vol. 34, Issue 6, Jun 2001, pg. 28.

[199] Ceruti, M. and Thuraisingham, B., Dependable Objects for Databases, Middleware and Methodologies: A Position Paper, *Proceedings of the 5th International Workshop on Object-Orient Real-Time Dependable Systems, IEEE*, Nov 1999.

[200] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg. 17 &19.

[201] Bernstein, P., Middleware*, Communications of the ACM, Vol. 39, Issue 2*, Feb 1996, pg 91.

identify a component's function and the way to invoke its behavior.[202]

Internet2® has been working in the standardization arena. Some projects include research and education middlewares. The group, MACE (Middleware Architecture Committee for Education), is trying to promote interoperability in the area of security and directories. They are trying to promote good-practice documents, devise pilot projects, and recommend technical standards that will support academic and administrative needs. Another project called SPARC (for a group researching upper atmospheric and space physics) is working on real-time collaboration tools.[203]

Currently, the off-the-shelf middleware technologies focus on some of the following factors that should be standardized:

- Inter- and intra-process communication - "a distributed application is likely to contain a mix of components that execute in a single thread of control, in different threads of control (but in the same process), and in different processes, some of which will reside on different machines."
- Features of software connectors – the middleware technology provides ability for two processes to exchange data. Additional connector features include event routing (such as broadcast, multicast, point-to-point), filtering, and registration.
- Platform and language support – a software architecture assembled out of components that supports multilingual and multi-platform applications.
- Communication method – a middleware should use most of the communication modes (RPC, message-passing, passing object references, shared memory, etc.).

---

[202] Lewandowski, S., Framework for Component-Based Client\Server Computing, *ACM Computing Survey (SCUR), Vol. 30, Issue 1*, Mar 1998, pg 10.

[203] Internet2, http://middleware.internet2.edu/internet2-and-sparc.html and http://middleware.internet2.edu/MACE/, found 2/14/2002.

- Ease of integration and use – is a way of integrating the middleware technology into the implementation infrastructure so that the user puts forth minimum effort.
- Multiple instances in an application – in a distributed system, an application can get data from several sources which helps prevent bottlenecks and increases scalability.
- Support for dynamic change – support for run-time modification assisting an application to be flexible.
- Performance – this is especially important in real-time applications and users expect it. [204]

ISO introduced in 1998 a standard for a middleware framework to encompass synchronous communication, distributed media resources, and seamless integration of data and process from disparate applications. It is called PREMO (PResentation Environments for Multimedia Objects). It was originally created as a new computer graphics standard using OO programming. Now the area is open to further develop it to include other medias (such as audio and video). It can also include the construction of distributed systems that involve generation, processing, and presentation of the data. [205]

The Securities Industry Middleware Council (SIMC) has also become involved with standardization. Their mission is to enhance the Securities Industry's capabilities to interoperate as a global electronic marketplace where any size companies, anywhere in the

---

[204] Dashofy, E., Medvidovic, N., and Taylor, R., Using Off-The-Shelf Middleware to Implement Connectors in Distributed Software Architectures, *Proceedings of the 1999 International Conference on Software Engineering (ICSE '99)*, May 1999, pg 7.

[205] Duke, D. and Herman, I., A Standard for Multimedia Middleware, *ACM Multimedia'98*, Aug 1998, pg 381.

world, could meet and conduct business with each other. Middleware will be a key to facilitate the communication in this ever-growing environment. [206]

A problem exists when the competitive middleware products have overlapping standard activities or implement the same technologies inconsistently. These problems are often caused when one tries to combine more than one middleware to solve a problem; it can lead to confusion and uncertainty in the effort to create a system with black-box type middleware. The solution is that middlewares become standardized on common conventions, structures, and documentation and modularized so that multiple middlewares may be used together. [207] Another solution would be for middleware products to offer a large assortment of core services (to include session management, queue management, load balancing, security, memory management, compression, directory services, and many more) in a modular form. This middleware should support many programming languages, network protocols, and OS platforms.[208]

When selecting the middleware product or products that will be used to accomplish the desired tasks, carefully research is essential. Each new middleware product can have a steep learning curve and may require many man-hours to incorporate it into a new or existing system. If there were middleware metrics, this job would require less time. Unlike software engineering, source lines of code are not a good enough metric to estimate

---

[206] SIMC Organization website, http://www.simc-inc.org/mission3.htm found 3/10/2002.

[207] Brown, A., Mastering the Middleware Muddle, *IEEE Software*, Jul/Aug 1999, pg 18.

[208] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg. 19.

productivity. [209] Standardization of middleware product's structures would significantly help in the area of selecting the correct middleware to fulfill the requirements and allow for better estimation of the integration of the middleware into the system. One author said this the best by stating: "The only thing that's going to survive in the middleware world is a standards-based approach that will run on everything." [210]

Some middleware metric issues that need to be addressed in the future are:

- Modifiability/Maintainability – How easy is it to make changes to the middleware to resolve problems and to accommodate updates in other parts of the system?
- Understandability/Usability – How easy is it to understand the use of the middleware, its inputs/outputs, communication protocols, wrappers requirements, language and platform requirements, etc.?
- Adaptability/Abstraction/Reusability/Portability – How able is the middleware to conform to the application, network, and other programs used in the system?
- Reliability/Correctness/Integrity of consistency – How well does the middleware provide assurance that the data is transferred accurately and within the time frame requirements?
- Security/Authentication/Integrity/Confidentiality – How well does the middleware provide a secure condition whereby the authorized user can perform its operations and access the needed data and other users cannot?

---

[209] Dousette, P., Danesh, A., and Jones, M., Command and Control using World Wide Web Technology, *Proceedings of the ACM SIGAda Annual International Conference on Ada Technology, Vol. 18, Issue 6*, Nov 1997, pg 214.

[210] Milojičić, D., Middleware's Role, Today and Tomorrow, *IEEE Concurrency, Vol. 7, Issue 2*, Apr/Jun 1999, pg 76.

- Integration – How well does the middleware operate with other parts of the system?

4.2 Future Trends of Middlewares

Middlewares are here to stay and it is estimated that the middleware and businessware markets will exceed $5 billion by 2005.[211] Middlewares are becoming more adaptable to programming languages and multiple OS platforms, but there are more changes needed. The literature contains several references to anticipated changes in the middleware infrastructure. Here is the information on these changes expected in the near future.

The general approach today is to develop a middleware using the black-box philosophy. This method hides the details of the operation from the user, application, and programmer. There is increasing evidence that the use of the black-box position is becoming untenable. OMG™ has recently added an interface to CORBA® to provide for more open applications. The Portable Object Adapter (POA) is another effort toward a more open design. In addition, many ORB vendors are reluctantly showing selected features of the underlying system to the user. Unfortunately, this may cause problems with portability of their products. [212]

According to another author, two major areas of the computing environment are growing: web and multitiered distributed systems. There have been major improvements in middlewares for connecting databases and the Internet. Middlewares (ORBs, TP monitors, middle-tier solutions) are providing the foundation for the future of the distributed

---

[211] Middleware and Businessware Markets are Meeting 21st Century Criteria, http://yahoo.bitpipe.com/data/detail?id=1004995910_239&type=RES&x=536047859, found 2/15/2002.

[212] Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr. 2000, pg 166.

computing. New and better tools are being produced to support these products. The capabilities of mature products have been better documented. [213]

The National Science Foundation (NSF) Middleware Initiative (NMI) is interested in creating a new middleware. This middleware will be used to:

a) "Facilitate scientific productivity,
b) Increase research collaboration through shared data, computing, code, facilities and applications
c) Support the education enterprise
d) Encourage the participation of industry partners, government labs, and agencies for more extensive development and wider adoption and deployment
e) Establish a level of persistence and availability so that other applications developers and disciplines can take advantage of the middleware
f) Encourage and support the development of standards and open source approaches
g) Enable scaling and sustainability to support the larger research and education communities." [214]

Several authors indicate that adaptive agents or reflective middlewares are a future trend. Adaptive agents can act as an impedance match between the network and the objects connected to it. [215] An adaptive middleware platform provides tools that are computational resource aware, management policies conscious, and understand how the system performs resource allocation.[216]

A reflective system can perform an inspection on itself and then make adaptive changes based on that inspection. [217] These changes are performed on software and hardware

---

[213] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, (http://www.dbmsmag.com/9709d14.html), found 3/29/2001, pg 7.

[214] NSF Middleware Initiative (NMI), http://www.nsf-middleware.org/Outreach/Announcements.htm/, found 2/14/2002.

[215] Kleinrock, L., Breaking Loose, *Communications of the ACM*, Sep 2001, pg 45.

[216] Duran, H. and Blair, G., A Resource Management Framework for Adaptive Middleware, *Proceeding of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Mar 2000.

mechanisms without any intervention of the applications or end users. [218] With a reflective system, it is "crucial to address the issue of integrity of the underlying middleware platform given the level of openness and extensibility inherent in reflective technology." It is essential that the technologies that are created, can limit the amount of change by employing solutions emerging from the component community such as component frameworks. [219]

Re-configurable middleware can cope with many fluctuations in the computing environment. This middleware can support system evolution as requirements change over time. One way this is accomplished is by maintaining the system status so that changes can be detected.[220] Another way to achieve this is through the use of an extensive library of components. The middleware can be compiled as one configuration and be changed with the use of these libraries to another. [221]

Another area of advancement is in the mobile or wireless middleware area. Within the next few years, universal devices will choose and access the closest, finest quality, and lowest cost wireless network. This wireless network will be competing with the "wired" networks. "With an increase of frequency allocation, improvement in semiconductor

---

[217] Duran, H. and Blair, G., A Resource Management Framework for Adaptive Middleware, *Proceeding of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Mar 2000.

[218] Wang, N., Kircher, M., and Schmidt, D., Applying Reflective Middleware Techniques to Optimize a QoS-enabled CORBA Component Model Implementation, *Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC)*, Oct 2000, pg 2.

[219] Eliassen, F., et al, Next Generation Middleware: Requirements, Architecture, and Prototypes, *7th IEEE Workshop on Future Trends on Distributed Computing Systems*, Dec 2000.

[220] Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr. 2000, pg 164.

[221] Eliassen, F., et al, Next Generation Middleware: Requirements, Architecture, and Prototypes, *7th IEEE Workshop on Future Trends on Distributed Computing Systems*, Dec 2000.

technology, and more efficient coding of information over wireless channels, mobile and wireless networks will become the networks of choice from most users and applications, making wired networks relics of the past." The ATM (Asynchronous Transfer Mode) Forum (a worldwide consortium of companies promoting ATM applications) will soon be releasing the final standards of this type of wireless network protocols.[222]

According to another paper, the next-generation middleware architecture will require the following factors:

- "The increasing importance of component integration, as opposed to programming;
- New paradigms for distributed computing based on mobile code and mobile agent technologies;
- Development of context-aware smart environments; and
- Multimedia-based real-time interactions in groupware and collaboration systems."

To accomplish these factors, the middleware will need a design that supports policy-driven integration of the components using system-level services and resources. [223]

Another issue that must be covered by the next-generation middleware products is security. The distributed systems are more complex today than ever before and require more protection against viruses, eavesdroppers, and other security issues. With the mobile computing becoming the standard mode of operation, security issues will be especially important (see section 3.7.2 above).

---

[222] Varshney, U. and Vetter, R., Emerging Mobile and Wireless Networks, *Communications of the ACM, Vol. 43, Issue 6*, Jun 2000, pg 77.

[223] Tripathi, A., Challenges Designing Next-Generation Middleware Systems, *Communications of the ACM, Vol. 45, No. 6*, Jun 2002, pg 39.

5. CONCLUSION

The first part of this paper discusses the true definition of middleware. As was stated, middleware is the software that enables application(s) to interact or communicate with other applications, networks, hardware, and/or operating systems. This software assists programmers by relieving them of complex lines of code needed to connect applications with distributed systems. It provides tools for improving quality of service (QoS), network fault tolerance, security (authentication, confidentiality, certification, integrity and access controls), message passing, file services, directory services, etc. in a transparent way to the user.

The second and most important issue of this thesis is the classification of these different types of middleware. The categorizations proposed by other authors were discussed. A new categorization was then presented in which middlewares are divided into two major grouping (Integration and Applications). These two major classifications were then subdivided into categories based on the mode of operation or the support given to the applications. Figure 1 above displays a breakdown of the middleware categorization schema. Figure 9 gives another view as to the interpretation of this type of categorization.

As part of the categorization of these middlewares, a description of how different middlewares are used was discussed in chapter 3. The strengths and weaknesses of these middlewares were elaborated to clarify that no one middleware can provide its services without having some negative aspect.

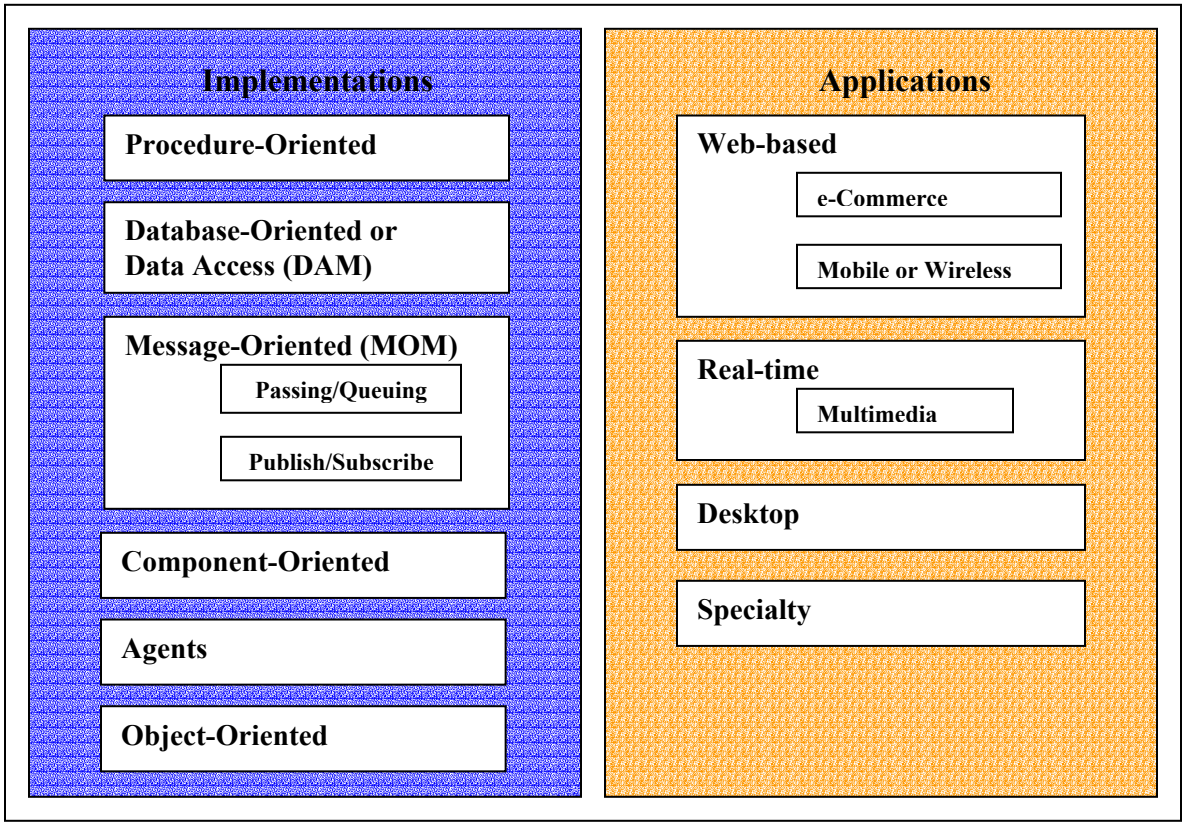

Figure 9 - Categorization of Middlewares

Other issues covered in this thesis included the standardization of middlewares. This is very important for the future because the pace of production of new software products is ever increasing. There is always a need for middleware products that are easy to understand, use, and that provide for application required assistance with minimal testing. One author stated the standardization issues quite clearly. "That's the thing that's most important; it doesn't matter how powerful the middleware is or how many great things it can do if no one can figure out how to use it."[224]

[224] Milojičić, D., Middleware's Role, Today and Tomorrow, *IEEE Concurrency, Vol. 7, Issue 2*, Apr/Jun 1999, pg 71.

This thesis was concluded with the future trends for middleware products. Chapter 4 has a record of information found about the future of middlewares. Although, no one can really tell the future, this chapter has up-to-date information and the trends toward these avenues appears to be correct.

APPENDIX

MIDDLEWARE DEFINITIONS

Date	Quotes
Oct 1993	Software insulates applications from the complexity of low-level system programming. It provides the necessary standards, components, and universal interfaces that must exist as an abstraction bridge between low-level software/hardware implementation and applications components.[225]
Feb 1996	A layer above the OS and networking software and below industry-specific application.[226]
Jun 1996	A word introduced for designating those areas of software services that "sit" above the traditional network protocols and provide means for extending services commonly available on the Internet to enclose higher layers of abstraction useful in composing new and old programs in complex software ensembles based on features like those found in standard platforms for distributed objects.[227]
Dec 1996	A set of services and protocols that exist just below the applications themselves and provide a set of common application services.[228]
Jul 1997	Middleware is like the kitchen wall: the pipes and wires bringing electricity, water, gas to and from the kitchen. ... Middleware provides services that connect the user interface, applications, data, and delivery system.[229]
Sep 1997	Just as with a sandwich, it is what is in the middle that matters.[230]
Nov 1997	A valuable software layer/system, which allows users to develop large complex, distributed applications without having to deal with details of the underlying networking and operating system.[231]

[225] A. Filarey, et al, *Software First*, Conference Proceedings on TRI-Ada '93, Oct 1993, pg. 90 – 101.

[226] Bernstein, P., Middleware*, Communications of the ACM, Vol. 39, Issue 2*, Feb 1996, pg 88.

[227] Ciancarini, P., Coordination Models and Languages as Software Integrators, *ACM Computing Surveys (CSUR), Vol. 28 Issue 2*, Jun 1996, pg 301.

[228] D. Clark and J. Pasquale, *Strategic Directions in Networks and Telecommunications*, ACM Computing Surveys (CSUR), Vol. 28 Issue 4, Dec 1996.

[229] M. Benda, *Middleware: Any Client, Any Server*, IEEE Internet Computing, Vol. 1, Issue 4, Jul/Aug 1997, pg 94 - 96.

[230] Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997, (http://www.dbmsmag.com/9709d14.html), found 3/29/2001, pg 1.

[231] G. Parulkar, et al., *Middleware for Distributed Multimedia (Panel)*, Proceedings of the 5th ACM International Conference on Multimedia, Nov 1997, pg. 347.

Date	Quotes
May 1999	Middleware has also become an important integration tool as an increasing number of companies – due to mergers, acquisitions, and infrastructure upgrades – try to assimilate multiple systems and applications.[232]
May 1999	A potentially useful tool when building software connectors. First, it can be used to bridge thread, process and network boundaries. Second, it can provide pre-built protocols for exchanging data among software components or connectors. Finally, some middleware packages include features of software connectors such as filtering, routing, and broadcast of messages or other data.[233]
Jul 1999	One of the most complex and confusing technology areas is *middleware* – the cornerstone required to build enterprise-scale distributed systems.[234]
Sep 1999	Software that is used to move information from one program to one or more other programs in a distributed environment, shielding the developer from dependencies on communication protocols, operating systems, and hardware platforms. [235]
Feb 2000	The major role of the middleware is to handle user authorization and multiple access, resolve differences among various computers involved and carry on remote method calls.[236]
May 2001	A software technology that enables the modular connection of distributed software.[237]
Feb 2002	Software that connects two otherwise separate applications.[238]
Feb 2002	Separate products that serve as the glue between two applications. ... Middleware is sometime called the plumbing because it connects two sides of an application and passes data between them.[239]

[232] Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg. 18.

[233] Dashofy, E., Medvidovic, N., and Taylor, R., Using Off-The-Shelf Middleware to Implement Connectors in Distributed Software Architectures, *Proceedings of the 1999 International Conference on Software Engineering (ICSE '99)*, May 1999, pg 7.

[234] A. Brown, *Mastering the Middleware Muddle*, IEEE Software, Jul/Aug 1999, pg 18 – 21.

[235] Campbell, A., Coulson, G., and Kounavis, M., Managing Complexity: Middleware Explained, *IT Pro*, Sep/Oct 1999, pg 27.

[236] N. Serbedżija, *Developing Middleware for Web-aware Systems: Lessons Learned*, Proceedings of the Austrilasian Computer Science Conference, Feb 2000.

[237] Astley, M., Sturman, D., et al, Customizable Middleware for Modular Distributed Software, *Communication of the ACM, Vol. 44, Issue 5*, May 2001, pg 101.

[238] Webopedia Website: http://www.webopedia.com/TERM/M/middleware.html , found 2/20/2002.

[239] Webopedia Website: http://www.webopedia.com/TERM/M/middleware.html , found 2/20/2002.

Date	Quotes
Jun 2002	Middleware provides an abstract interface that gives an application developer a uniform view of low-level operating system and networks. [240]

[240] Agha, G., Adaptive Middleware, *Communication of the ACM, Vol. 45, No. 6*, Jun 2002, pg 31.

REFERENCES

Agha, G., Adaptive Middleware, *Communication of the ACM, Vol. 45, No. 6*, Jun 2002, pg 31 – 32.

Altinel, M., et al, DBIS-Toolkit: Adaptive Middleware for Large Scale Data Delivery, *Proceedings of the 1999 International Conference on Management of Data (SIGMOD '99), Vol. 28, Issue 2,* May 1999, pg 544 - 546.

Arpirez, J., et al, Technical Papers: WebODE, *Proceeding of the International Conference on Knowledge Capture*, Oct 2001, pg 6 – 13.

Astley, M. and Agha, G., Customization and Composition of Distributed Objects: Middleware Abstractions for Policy Management, *ACM SIGSOFT Software Engineering Notes, Proceeding of the ACM SIGSOFT 6th International Symposium on Foundations of Software Engineering,,Vol. 23, Issue 6*, Nov 1998, pg 1 - 9.

Astley, M., Sturman, D., et al, Customizable Middleware for Modular Distributed Software, *Communication of the ACM, Vol. 44, Issue 5*, May 2001, pg 99 - 107.

Bacon, J. and Moody, K., Toward Open, Secure, Widely Distributed Services, *Communication of the ACM, Vol. 45, No. 6*, Jun 2002, pg 59 – 64.

Baru, C., XML-based Information Mediation for Digital Libraries, *Proceeding of the 4th ACM Conference on Digital Libraries*, Aug 1999, pg 214 – 215.

Bellavista, P., Corradi, A., and Stefanelli, C., Mobile Agent Middleware for Mobile Computing, *Computer, Vol. 34, Issue 3*, Mar 2001, pg 73 – 81.

Benda, M., The Architecture of Global Access, *IEEE Internet Computing*, Jan/Feb 1997, pg 78 –80.

Benda, M., Middleware: Any Client, Any Server, *IEEE Internet Computing, Vol. 1, Issue 4*, Jul/Aug 1997, pg 94 - 96.

Bernstein, P., Middleware*, Communications of the ACM, Vol. 39, Issue 2*, Feb 1996, pg 86 – 98.

Blair, G., et al, The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms, *IFIP/ACM International Conference on Distributed Systems Platforms*, Apr 2000, pg 164 - 184.

Britton, C., *IT Architectures and Middleware - Strategies for Building Large, Integrated Systems*, Addison-Wesley Publishing, ISBN 0201709074, Apr 2001.

Bronsard, F., et al, Toward Software Plug and Play, *ACM SIGSOFT Software Engineering Notes, Proceedings of the 1997 Symposium on Software Reusability, Vol. 22, Issue 3*, May 1997, pg 19 – 29.

Brown, A., Mastering the Middleware Muddle, *IEEE Software*, Jul/Aug 1999, pg 18 – 21.

Brunsch, D., O'Ryan, C., and Schmidt, D., Designing an Efficient and Scalable Server-side Asynchrony Model for CORBA, *ACM SIGPLAN Notices, Vol. 36, Issue 8*, Aug 2001, pg 223 – 229.

Middleware and Businessware Markets are Meeting 21st Century Criteria, http://yahoo.bitpipe.com/data/detail?id=1004995910_239&type=RES&x=536047859, found 2/14/2002.

Campbell, A., Coulson, G., and Kounavis, M., Managing Complexity: Middleware Explained, *IT Pro*, Sep/Oct 1999, pg 22 – 28.

Caron, O., Carré, B., and Debrauwer, L., An Original View Mechanism for the CORBA Middleware, *IEEE Proceedings of the Technology of Object-Oriented Language and Systems (TOOLS 33)*, Jun 2000.

Carothers, C., et al, Visualizing Parallel Simulations in Network Computing Environments, *Proceedings of the Conference on Winter Simulation*, Dec 1997, pg 110 – 117.

Carzaniga, A., et al, Issues in Supporting Event-Based Architectural Styles, *Proceedings of the 3rd International Workshop on Software Architecture*, Nov 1997, pg 17 – 20.

Carzaniga, A., et al, Achieving Scalability and Expressiveness in an Internet-Scaled Event Notification Service, *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*, Jul 2000, pg 219 – 227.

Ceruti, M. and Thuraisingham, B., Dependable Objects for Databases, Middleware and Methodologies: A Position Paper, *Proceedings of the 5th International Workshop on Object-Orient Real-Time Dependable Systems, IEEE*, Nov 1999.

Charles, J., Middleware Moves to the Forefront, *Computer*, May 1999, pg 17 - 19.

Changing Face of Middleware, http://yahoo.bitpipe.com/data/detail?id=968849360_790&type=RES&x=970187924, found 2/14/2002

Chiang, C., A Distributed Object Computing Architecture for Leveraging Software Engineering Systems, *Proceeding of the 16th ACM SAC2001 Symposium on Applied Computing*, Mar 2001, pg 653 – 657.

Christophides, V., Cliet, S., and Simèon, J., On Wrapping Query Languages and Efficient XML Integration, *Proceedings of the 2000 ACM SIGMOD on Management of Data, Vol. 29, Issue 2*, May 2000, pg 141 - 152.

Chuang, S., Securing ATM Networks, *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, Jan 1996, pg19 – 30.

Ciancarini, P., Coordination Models and Languages as Software Integrators, *ACM Computing Surveys (CSUR), Vol. 28 Issue 2*, Jun 1996, pg 300 - 302.

Clark, D. and Pasquale, J., Strategic Directions in Networks and Telecommunications, *ACM Computing Surveys (CSUR), Vol. 28 Issue 4*, Dec 1996, pg 679-690.

Conrad, C. and Stiller, B., The Design of an Application Programming Interface for QoS-based Multimedia Middleware, *Proceeding of the 22nd IEEE Conference on Local Computer Networks (LCN '97)*, Nov 1997, pg 274 - 283.

Costa, F. and Blair, G., Integration Meta-Information Management and Reflection in Middleware*, Proceedings of the IEEE International Symposium on Distributed Objects and Applications (DOA)*, Sep 2000.

Coulson, G., A Configerable Multimedia Middleware Platform, *IEEE Multimedia*, Jan/Mar 1999, pg. 62 – 76.

Choosing CT Middleware, http://yahoo.bitpipe.com/data/detail?id=989602863_673&type=RES&x=738520556, found 2/14/2002.

DataMirror Whitepapers, Managing your Data the XML Way: Data Transformation, Exchange and Integration, http://www.datamirror.com/resourcecenter/download/dbxmlvision/pdfs/dbxmltransform/XML_solutions.pdf, found 2/14/2002.

Dashofy, E., Medvidovic, N., and Taylor, R., Using Off-The-Shelf Middleware to Implement Connectors in Distributed Software Architectures, *Proceedings of the 1999 International Conference on Software Engineering (ICSE '99)*, May 1999, pg 3 - 12.

Degenaro, L., et al, A Middleware System which Intelligently Caches Query Results, *IFIP/ACM International Conference on Distributed Systems Platforms*, 2000, pg 24 – 44.

Di Nitto, E. and Rosenblum, D., Exploiting ADLs to Specify Architectural Styles Induced by Middleware Infrastructures, *ACM Proceedings of the 1999 International Conference on Software Engineering*, May1999, pg 13 – 22.

Ding, Y., et al, RAJA, *Proceedings of the 5th International Conference on Autonomous Agents*, May 2001, pg 332 - 339.

Dousette, P., Danesh, A., and Jones, M., Command and Control using World Wide Web Technology, *Proceedings of the ACM SIGAda Annual International Conference on Ada Technology, Vol. 18, Issue 6*, Nov 1997, pg 212 – 214.

Duke, D. and Herman, I., A Standard for Multimedia Middleware, *ACM Multimedia'98*, Aug 1998, pg 381 – 390.

Duran, H. and Blair, G., A Resource Management Framework for Adaptive Middleware, *Proceeding of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Mar 2000.

Eliassen, F., et al, Next Generation Middleware: Requirements, Architecture, and Prototypes, *7th IEEE Workshop on Future Trends on Distributed Computing Systems*, Dec 2000.

Emmerich, W., Distributed Objects, *Proceedings of the 1999 International Conference on Software Engineering*, May 1999, pg 665 – 666.

Emmerich, W., Schwarz, W., and Finkelstein, A., Markup Meets Middleware, *7th IEEE Workshop on Future Trends in Distributed Computing Systems*, Dec 1999.

Emmerich, W., Software Engineering and Middleware: A Roadmap, *ACM Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 119 - 129.

Engerman, G. and Kearney, L., Effective Use of Wireless Data Communications, *International Journal of Network Management, Vol. 8, Issue 1*, Feb 1998, pg 2 – 11.

Engineering Back Office, http://yahoo.bitpipe.com/data/detail?id=999539692_936&type=RES&x=253484853, found 2/14/2002.

Fagin, R., Fuzzy Queries in Multimedia Database Systems, *ACM*, Mar 1998.

Fayad, M., Introduction To The Computing Surveys' Electronic Symposium on Object-Oriented Application Frameworks, *ACM Computing Surveys (CSUR)*, Mar 2000.

Fenestrae Mobile Data Server, http://yahoo.bitpipe.com/data/detail?id=1004968144_500&type=RES&x=1335401392 found 2/14/2002.

Filarey, A., et al, Software First, *Conference Proceedings on TRI-Ada '93*, Oct 1993, pg 90 – 101.

Fraternali, P., Tools and Approaches for Developing Data-Intensive Web Applications: A Survey, *ACM Computing Survey (CSUR), Vol. 31, Issue 3*, Sep 1999, pg 227 – 263.

Fregonese, G., Zorer, A., and Cortese, G., Architecture Framework in Telecommunication Domain, *Proceedings of the 1999 International Conference on Software Engineering*, May 1999, pg 526 – 534.

Fuentes, L. and Troya, J., Toward an Open Multimedia Service Framework, *ACM Computing Surveys, Vol. 32, No. 1*, Mar 2000.

Geihs, K., Middleware Challenges Ahead, *IEEE Computer, Vol. 34, Issue 6*, Jun 2001, pg 24 – 31.

Gimenez, G. and Kim, K., A Windows CE Implementation of a Middleware Architecture Supporting Time-Triggered Message-Triggered Objects, *Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC01)*, Jun 2000.

Graham J. and Cepull, J., Early Adopters, An Internet 2 Middleware Project, *ACM Proceedings of the Conference on User Services: Building the Future*, Oct 2000, pg 86 - 91.

Hasselbring, W., Information System Integration, *Communications of the ACM, Vol. 29, Issue 2*, Jun 2000, pg 33 – 38.

Hecht, M., et al, OFTT: A Fault Tolerance Middleware Toolkit for Process Monitoring and Control Window NT Application, *Proceedings of the IEEE International Conference on Dependable Systems and Networks*, Jun 2000.

Heimbigner, D., Adapting Publish/Subscribe Middleware to Achieve Gnutella-like Functionality, *Proceedings of the 16th ACM SAC2001 Symposium on Applied Computing*, Mar 2001, pg 176 – 181.

Heuser, L., et al, Development of Distributed and Client/Server Object-Oriented Applications, *Proceedings of the 9th Annual Conference on Object-Oriented Programming Systems, Language, and Applications, Vol. 29, Issue 10*, Oct 1994, pg 317 – 323.

Hofmann, C., A Multi-Tier Framework for Accessing Distributed, Heterogeneous Spatial Data in a Federated Based EIS, *Proceedings of the 7th International Symposium on Advances in Geographic Information Systems*, Nov 1999, pg 140 – 145.

Hou, C., Han, C., and Chen, T., Communication Middleware and Software for QoS Control in Distributed Real-Time Environments, *COMPSAC '97 – 21st International Computer Software and Applications Conference*, Aug 1997.

Internet2, http://middleware.internet2.edu/, http://middleware.internet2.edu/multicampus/, http://middleware.internet2.edu/internet2-and-sparc.html, http://www.internet2.edu/presentations/20020201-camp-blatecky.ppt, and http://middleware.internet2.edu/MACE/ found 2/14/2002.

Issarny, V., Bidan, C., and Saridakis, T., Achieving Middleware Customization in a Configuration-Based Development Environment: Experience with the Aster Prototype, *IEEE*, 1997.

Issarny, V., Kloukinas, C., and Zarras, A., Systematic Aid for Developing Middleware Architectures, *Communication of the ACM, Vol. 45, No. 6*, Jun 2002, pg 53 – 58.

IT Pro, http://www.itprodownloads.com/filedownload?fileid=178578/, found 2/14/2002.

Jacobsen, H. and Günther, O., Middleware for Software Leasing Over the Internet, *Proceedings of the First ACM Symposium on Electronic Commerce*, Nov 1999, pg 87 – 95.

Jain, R., Anjum, F., and Umar, A., A Comparison of Mobile Agent and Client-Server Paradigms for Information Retrieval Task in Virtual Enterprises, *IEEE*, 2000.

Jeng, J., Towards a Universal Service-Computing Platform via Virtual Service Machine, *Proceedings of the 16th ACM SAC2001 Symposium on Applied Computing*, Mar 2001, pg 663 – 667.

Jeng, J., An Approach to Designing Reusable Service Frameworks via Virtual Service Machine, *ACM SIGSOFT Software Engineering Notes, Proceedings of SSR'01 on 2001 Symposium on Software Reusability, Vol. 26, Issue 3*, May 2001, pg 58 – 66.

Johnson, S., et al, Experiences with Group Communication Middleware, *Proceedings of the IEEE International Conference on Dependable Systems and Networks (DSN)*, Jun 2000.

Jones, C., et al, A Survey of Energy Efficient Network Protocols for Wireless Networks, *Wireless Networks, Vol. 7, Issue 4*, Sep 2001, pg 343 - 358.

Jung, D., Paek, K., and Kim, T., Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing, *IEEE International Workshops on Parallel Processing*, Sep 1999.

Kim, K., et al, A CORBA Service Enabling Programmer-Friendly Object-Oriented Real-Time Distributed Computing, *IEEE 5th International Workshop on Object-Oriented Real-Time Dependable Systems*, Nov 1999.

Kim, K., Ishida, M., and Liu, J., An Efficient Middleware Architecture Supporting Time-Triggered Message-Triggered Objects and an NT-based Implementation, *2nd IEEE International Symposium on Object-Orient Real-Time Distributed Computing*, May 1999.

Kleinrock, L., Breaking Loose, *Communications of the ACM*, Sep 2001, pg 41 – 45.

Kloukinas, C. and Issarny, V., Automating the Composition of Middleware Configuration, *Proceedings of the IEEE International Conference on Automation Software Engineering*, Sep 2000.

KnowledgeStorm, http://www.knowledgestorm.com/, headquartered in Atlanta, GA, found 2/20/2002.

Kon, F., et al, The Case for Reflective Middleware, *Communications of the ACM, Vol. 45, no. 6*, Jun 2002, pg 33 – 38.

Krishnamurthy, Y., et al, Integration of QoS-Enabled Distributed Object Computing Middleware for Developing Next-Generation Distributed Applications, *ACM SIGPLAN Notices, Vol. 36, Issue 8*, Aug 2001, pg 230 – 237.

Kuropka, D. and Weske, M., Transparent and Flexible Storage of Application Objects in CORBA Environments, *Addendum to the 2000 Proceedings of the Conference on Object-oriented Programming, Systems, Languages, and Applications*, Mar 2000, pg 169 – 170.

Kutlusan, A., et al, A Combat Management System Middleware Based on CORBA, *Proceedings of the IEEE International Symposium on Distributed Objects and Applications*, Sep 2000.

Lewandowski, S., Framework for Component-Based Client\Server Computing, *ACM Computing Survey (SCUR), Vol. 30, Issue 1*, Mar 1998, pg 3 - 27.

Lewis, T., Where is Client/Server Software Headed?, *IEEE Computer, Vol. 28, Issue 4*, Apr 1995, pg 49 - 55.

Lloyd, A., Dewar, D., and Pooley, R., Legacy Information Systems and Business Process Change, *Communications of the AIS*, Dec 1999, pg 1 – 41.

Li, B. and Nahrstedt, K., QualProbes: Middleware QoS Profiling Services for Configuring Adaptive Applications, *IFIP/ACM International Conference on Distributed System Platforms*, Apr 2000, pg 256 - 272.

LIME Open Source License, http://lime.sourceforge.net, found 6/30/2002.

Linthicum, D., Next Generation Middleware, *DBMS*, Sep 1997 (http://www.dbmsmag.com/9709d14.html), found 3/29/2001, pg 1-9.

Linthicum, D., Database-Oriented Middleware, *DM Review*, Nov 1999, (http://www.dmreview.com/editorial/dmreview/print_action.cmf?EdID=1560), found 3/29/2001, pg 1 - 12.

Malan, G., Jahanian, F., and Knoop, P., Comparison of Two Middleware Data Dissemination Services in a Wide-Area Distributed System, *Proceedings of the 17th International Conference of Distributed Computing System (ICDCS '97)*, May 1997, pg 411 - 419.

Marrón, P., Design of a Middleware Service for Scalable Wide Are Network Applications, *Proceedings of the IEEE International Symposium on Distributed Objects and Applications (DOA),* Sep 2000, pg 1 - 10.

McFall, C., An Object Infrastructure for Internet Middleware, *IEEE Internet Computing*, Mar/Apr 1998, pg 46 - 51.

McKinley, P., Malenfant, A., and Arango, P., Pavilion, *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, Nov 1999, pg 179 – 188.

Merriam-Webster, Webster's Ninth New Collegiate Dictionary, Springfield, Massachusetts, U.S.A., 1987.

Middleware and Businessware Markets are Meeting 21st Century Criteria, http://yahoo.bitpipe.com/data/detail?id=1004995910_239&type=RES&x=536047859, found 2/15/2002.

Mills, K., Introduction to the Electronic Symposium on Computer-Supported Cooperative Work, *ACM Computing Surveys (CSUR), Vol. 31, Issue 2*, Jun 1999, pg 105 - 115.

Milojičić, D., Middleware's Role, Today and Tomorrow, *IEEE Concurrency, Vol. 7, Issue 2*, Apr/Jun 1999, pg 70 - 80.

Milojičić, D., et al., Process Migration, *ACS Computing Surveys, Vol. 32, Issue 3*, Sep 2000, pg 241 - 299.

MIT Media Lab: Software Agents: Projects, May 2001, http://agents.media.mit.edu/projects/, found 2/30/2002.

Mobile Middleware for Wireless Data Devices, http://yahoo.bitpipe.com/data/detail?id=973011263_806&type=RES&x=186610175, found 2/14/2002.

Moeller, M., Middleware Vendors Eye Network Market, *IEEE Software*, Sep 1994, pg 116 – 118.

Mortazavi, M., On Framing Object Relationships to Improve QoS in Distributed Systems, *IEEE 5th International Workshop on Object-Oriented Real-Time Dependable Systems*, Nov 1999.

Narain, S., et al, Middleware for Building Adaptive Systems via Configuration, *ACM SIGPLAN Notices, Vol. 36, Issue 8*, Aug 2001, pg 188 – 195.

NSF Middleware Initiative (NMI), http://www.nsf-middleware.org/Outreach/Announcements.htm/, found 2/14/2002.

Nürnberg, P. and Ashman, H., What was the question? Reconciling open hypermedia and World Wide Web Research, *Proceeding of the 10th ACM Conference on Hypertext and Hypermedia*, Feb 1999, pg 83 – 90.

O'Ryan, C., et al, Applying a Scalable CORBA Event Service to Large-scale Distributed Interactive Simulation, *IEEE 5th International Workshop on Object-Oriented Real-Time Dependable Systems*, Nov 1999, pg 1 - 8.

O'Ryan, C., et al, The Design and Performance of a Pluggable Protocols Framework for Real-Time Distributed Object Computing Middleware, *IFIP/ACM International Conference on Distributed System Platforms*, Apr 2000, pg 372 – 393.

Othman, O. and Schmidt, D., Issues in the Design of Adaptive Middleware Load Balancing, *ACM SIGPLAN Notices, Vol. 36, Issue 8*, Aug 2001, pg 205 – 213.

Parulkar, G., et al, Middleware for Distributed Multimedia (Panel), *Proceedings of the 5th ACM International Conference on Multimedia*, Nov 1997, pg 347.

Pereira, C. (Panel Organizer), Middleware for Real-Time Distributed Objects: Needs and Requirements from Different Application Domains, *Proceedings of the 6th International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS' 01), IEEE Computer Society*, 2001.

Petriu, D., et al, Using Analytic Models for Predicting Middleware Performance, *ACM Proceedings on the 2nd International Workshop on Software and Performance*, Sep 2000, pg 189 - 194.

Phan, T., Guy, R., and Bagrodia, R., A Scalable, Distributed Middleware Service Archicture to Support Mobile Internet Applications, *Proceedings of the First Workshop on Wireless Mobile Internet*, Jul 2001, pg 27 – 33.

Picco, G., Murphy, A., and Roman, G., Developing Mobile Computing Applications with LIME, *Proceedings of the 22nd International Conference on Software Engineering*, Jun 2000, pg 766 – 769.

Pyarali, I., et al, Evaluating and Optimizing Thread Pool Strategies for Real-Time CORBA, *ACM SIGPLAN Notices, Vol. 36, Issue 8*, Aug 2001, pg. 214 – 221.

Pyarali, I., O'Ryan, C., and Schmidt, D., A Pattern Language for Efficient, Predictable, Scalable, and Flexible Dispatching Mechimisms for Distributed Object Computing Middleware, *3rd IEEE/IFIP International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, Mar 2000, pg 1 - 8.

Rackl, G., et al, MIMO - An Infrastructure for Monitoring and Managing Distributed Middleware Environments, *IFIP/ACM International Conference on Distributed System Platforms*, Apr 2000, pg 71 - 87.

Ranganathan, M., et al, Mobile Streams: A Middleware for Reconfigurable Distributed Scripting, *Proceedings for the 1st International Symposium on Agent Systems and Applications & 3rd International Symposium on Mobile Agents*, Oct 1998, pg 1 – 14.

Improving Reliability and Scalability in the Middle Tier of e-Business, http://yahoo.bitpipe.com/data/detail?id=983872839_224&type=RES&x=1733437713, found 2/14/2002.

Rodriguez-Martinez, M., and Roussopoulos, N., MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources, *ACM 2000 1-58113-218-2/00/05*, May 2000, pg 213 - 224.

Roman, G., Picco, G., and Murphy, A., Software Engineering for Mobility, *Proceedings of the Conference on the Future of Software Engineering*, May 2000, pg 241 – 258.

Rouff , C., and Robbert, M., Developing the Cooperative Mission Development Environment, *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, Nov 1997, pg 351 – 357.

Royce, W., Boehm, B., and Druffel, C., Employing UNAS Technology For Software Architecture Education at the University of Southern California, *11th Annual Washington Ada Symposium*, Jul 1994, pg 113 – 121.

Ruggaber, R., Seitz, J., and Knapp, M., Π^2 - a Generic Proxy Platform for Wireless Access and Mobility in CORBA, *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*, Jul 2000, pg 191 - 198.

Serbedżija, N., Developing Middleware for Web-aware Systems: Lessons Learned, *Proceedings of the Australasian Computer Science Conference*, Feb 2000.

Schmidt, D., Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects, *OOPSLA 2001 Conference*, Oct 2001.

Schmidt, D., Middleware for Real-Time and Embedded Systems, *Communications of the ACM, Vol. 45, No. 6*, Jun 2002, pg 43 – 48.

Shands, D., et al, Secure Virtual Enclaves, *ACM Transactions on Information and System Security (TISSEC), Vol. 4, Issue 2*, May 2001, pg 103 – 133.

Shokri, E. and Beltas, P., An Experiment with Adaptive Fault Tolerance in Highly-Constraint Systems, *IEEE 5th International Workshop on Object-Oriented Real-Time Dependable Systems*, Nov 1999.

SIMC Organization website, http://www.simc-inc.org/mission3.htm found 3/10/2002.

Slivinskas, G., Jensen, C., and Snodgrass, R., Adaptable Query Optimization and Evaluation in Temporal Middleware, *Proceedings of the 2001 ACM SIGMOND International Conference on Management of Data on Management of Data, May 2001, Vol. 30 Issue 2*, May 2001, pg 127 – 138.

Srivastrava, M., Smart Kindergarten, *The 7th Annual International Conference on Mobile Computing and Networking 2001*, Jul 2001, pg 132 – 138.

Stillerman, M., Marceau, C., and Stillman, M., Intrusion Detection for Distributed Applications, *Communications of the ACM, Vol. 42, Issue 7*, Jul 1999, pg 62 – 69.

Thompson, C., et al, Intermediary Architecture: Interposing Middleware Object Services between Web Client and Server, *ACM Computing Surveys (CSUR)*, Jun 1999.

Thompson, J., Avoiding a Middleware Muddle, *IEEE Software*, Nov/Dec 1997, pg 92 - 96.

Tripathi, A., Challenges Designing Next-Generation Middleware Systems, *Communications of the ACM, Vol. 45, No. 6*, Jun 2002, pg 39 – 42.

Umar, A., et al, A Knowledge-based Decision Support Workbench fro Advanced E-commerce, IEEE, 2000.

Uramoto, N. and Maruyama,H., InfoBus Repeater: A Secure and Distributed Publish/Subscribe Middleware, *1999 IEEE International Workshops on Parallel Processing*, Sep 1999.

Varshney, U. and Vetter, R., Emerging Mobile and Wireless Networks, *Communications of the ACM, Vol. 43, Issue 6*, Jun 2000, pg 73 – 81.

Venkatasubramanian, N., Safe 'Composability' of Middleware Services, *Communications of the ACM, Vol. 45, No. 6*, Jun 2002, pg 49 - 52.

Wang, N., Kircher, M., and Schmidt, D., Applying Reflective Middleware Techniques to Optimize a QoS-enabled CORBA Component Model Implementation, *Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC)*, Oct 2000, pg 1 - 8.

WAP Forum, http://www.wapforum.org/, found 6/10/2002.

Webopedia Website: http://www.webopedia.com/TERM/M/middleware.html, found 2/20/2002.

Whetten, B., Message-Based Computing: The Fourth Wave of Integration, http://www.messageq.com/communications_middleware/whetten_1.html, found 2/14/2002.

Wiederhold, G., et al, OntoAgents – a Project in the DARPA DAML Program, http://www-db.stanford.edu/Ontoagents, 1/29/2002.

Wiil, U., Nürnberg, P., and Leggett, J., Hypermedia Research Directions: An Infrastructure Perspective*, ACM Computing Survey (CSUR), Vol. 31, Issue 4*, Dec 1999, pg 1 – 9.

Wimmers, E., et al, Using Fagin's Algorithm for Merging Ranked Results in Multimedia Middleware, *Proceedings of the IECIS International Conference on Cooperative Information Systems*, Sep 1999.

Xingshe, Z. and Xiaodong, L., Design and Implementation of CORBA Security Service, *IEEE Technology of Object-Orient Languages and Systems (TOOLS 33)*, Jun 2000.

Yellin, D., Stuck in the Middle, *ACM SIGPLAN Notices*, Aug 2001, pg 175 – 180.

Zhou, J., Zhou, M., and Wu, Q., An Agent Framework Based on Distributed Object, *IEEE Technology of Object-Orient Languages and Systems (TOOLS 33)*, Jun 2000.

CURRICULUM VITA

NAME: Toni A. Bishop (maiden name – Toni Ann Ayers)

PROGRAM OF STUDY: Computer Science, Systems Science Track

DEGREE AND DATE TO BE CONFERRED: Masters of Science, September 2002

SECONDARY EDUCATION:

Cave Spring High School, Roanoke, Virginia, June 1975.

COLLEGES ATTENDED:

Longwood College, Farmville Virginia, 23901,
Bachelor of Science Degree in Chemistry & Biology, May 1979.
Towson State University, Towson, Maryland, 21252,
Bachelor of Science Degree in Computer Science, January 1993.
Towson University, Towson, Maryland, 21252,
Masters of Science Degree in Computer Science, Sep 2002.

PROFESSIONAL POSITIONS HELD:

Analytical Chemist, USAEHA, APG, MD, 21010, March 1980 – July 1989.

Analytical Chemist, USACHPPM, APG, MD, 21010, July 1989 – March 2001.

PUBLICATIONS:

Bishop, R., Ayers, T., and Rinehart, D., The Use of a Solid Sorbent as a Collection Medium for TNT and RDX Vapors, *AIHA Journal, Vol. 42, No. 8*, August 1981, pg 586 – 589.

Bishop, R., Ayers, T., and Esposito, G., A Gas Chromatographic Procedure for the determination of Airborne MDI and TDI, *AIHA Journal, Vol. 44, No. 3*, March 1983, pg 151 – 155.