2002

Derivation of the required elements for a definition of the term middleware

Maya Mathew

Derivation of the Required Elements for a Definition of the Term Middleware

By

Maya Mathew

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

Rochester Institute of Technology

B. Thomas Golisano College

Of

Computing and Information Sciences

May 2002

Rochester Institute of Technology

B. Thomas Golisano College
Of
Computing and Information Sciences

Master of Science in Information Technology

Thesis Approval Form

Student Name: Maya Mathew

Thesis Title: Derivation of the Required Elements for a Definition of the Term Middleware

Thesis Committee

Name	Signature	Date
Prof. William Stratton Chair		7/2/02
Prof. Andy Phelps Committee Member		7/2/02
Prof. Jeff Lasky Committee Member		June 17, 2002

Thesis Reproduction Permission Form

Rochester Institute of Technology

B. Thomas Golisano College
Of
Computing and Information Sciences

Derivation of the Required Elements

For a Definition of the Term Middleware

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

Date: 06/11/2002 Signature of Author: _____

TABLE OF CONTENTS

Abstract

Thirteen contemporary definitions of Middleware were analyzed. The definitions agree that any software that can do the following should be classified as Middleware (1) provide service that provides transparent application-to-application interaction across the network, (2) act as a service provider for distributed applications, and (3) provide services that are primarily used by distributed applications (e.g., RPCs, ORBs, Directories, name-resolution services, etc.) Most definitions agree that Middleware is that level of software required to achieve platform, location, and network transparency. There is some discrepancy about the OSI levels at which middleware operates. The majority of definitions limit it to levels 5, 6, and 7. Additionally, almost half of the definitions do not include database transparency as something achieved by Middleware, perhaps due to the ambiguous classification of ODBC and JDBC as software. Assuming that the number of times a service is mentioned, the majority of the definitions rank services associated with legal access to an application as core to Middleware, along with valid, standardized APIs for application development as core to the definition of middleware.

I. Overview

There is no doubt that Middleware is a big business today. Consider the analysis completed by Altman and Schulte:

> The market for integration middleware is growing quickly, both in terms of revenue and in the number of vendors and products that participate in the market. Product license revenue for integration broker suites will grow to almost $1 billion in 2000, up from $240 million in 1998 and $490 million in 1999. Counting the related support and service business, the total revenue for the integration broker middleware vendors is almost twice these figures. This rapid growth reflects the widespread need to integrate applications and the improve the effectiveness of off-the-shelf middleware to reduce the time and effort required to accomplish the goal of integration.
>
> (Altman & Schulte, 2000)

But what exactly is Middleware? As we shall see in this paper, many sources offer definitions for the term. Unfortunately, these definitions vary in both breadth and scope. So many different definitions have been offered, in fact, that, with considerable sarcasm, Information Week states that Middleware can be defined as "(1) a hodgepodge of software technologies; (2) a buzzword; (3) a key to developing client/server applications." (cited in Orfali, Harkey & Edwards, 1999, p. 44) and Clara H. Parkes, the editor of DBMS Magazine, asserts that Middleware "is now everywhere and everything, so pervasive that it has passed into the realm of a non-entity". (cited in Orfali, Harkey & Edwards, 1999, p. 44)

Given the ubiquity of the term, we undertake in this paper to review many of these definitions in order to seek out the commonalities among them and to craft a more rigorous definition of the term. In what follows, we will capitalize the word Middleware to emphasize our efforts to define it more completely and succinctly.

We have collected several published definitions of the term Middleware. We have also created a set of questions that we feel are related to the general idea of Middleware should be. The questions are:

- Should Middleware provide software that provides or supports transparent and "seamless" distributed environments?
- Where should Middleware be required to provide transparency?
- At what layers of the OSI model does (or should) Middleware operate?
- What services must be supplied by Middleware?

Presumably, if a majority of the definitions we have collected provide the same, or similar answers, to these questions, it will strengthen a common definition of the term while helping us eliminate those aspects that are particular to one definition but not shared by all.

II. A Survey of Proposed Definitions of Middleware

After a rigorous and thorough discussion, the participants of the Middleware workshop held at the International Center for Advanced Internet Research (ICIAR) in December 1998 used the OSI model to set boundaries on the term. They roughly characterize Middleware as "those services found above the transport (i.e., over TCP/IP) layer set of services, but below the application environment (i.e., below application-level APIs)". (Aiken et al, 2000, p. 4)

The workshop also concluded the following:

> Middleware may have components and services that only exist in the persistent infrastructure, but it will also have components that enable and support end-to-end (i.e. application to application or host to host) interaction across multiple autonomous administrative domains. A set of core persistent middleware services is required to support the development of a richer set of middleware services which can be aggregated or upon which applications will be based. The particular set of such services utilized by an application or process will be a function of the requirements of the application

> field or affinity group (e.g., network management or high energy physics applications) wishing to utilize the network or distributed data/computation infrastructure. Some basic and core middleware services include, but are not limited to: directories, name/address resolution services, security services (i.e., authentication, authorization, accounting, and access control), network management, network monitoring, time servers, and accounting. A second level of important middleware services, which builds upon these core set of services, may include accounting/billing, resource managers, single sign-on services, globally unique names, etc." (Aiken et al, p. 23-24)

The Software Engineering Institute at Carnegie Mellon defines Middleware as "connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network." (Bray, 1997) Under the SEI scheme, these enabling services are:

- Transaction processing (TP) monitors that provide tools and an environment for developing and deploying distributed applications.
- Remote Procedure Calls (RPCs) that enable the logic of an application to be distributed across the network. Program logic on remote systems can be executed as simply as calling a local routine.
- Message-Oriented Middleware (MOM) that provides program-to-program data exchange, enabling the creation of distributed applications. MOM is analogous to e-mail in the sense it is asynchronous and requires the recipients of messages to interpret their meaning and to take appropriate action.
- Object Request Brokers (ORBs) that enable the objects that comprise an application to be distributed and shared across heterogeneous networks.

Webopedia (Webopedia, 'middleware') defines the term in a similar fashion. Middleware is the "software that connects two otherwise separate applications", e.g., the products that link a database system to a web server. Here, the term refers to "separate products that serve as the glue between the two applications", and is,

therefore, "distinct from import and export features that may be built into one of the applications." It connects two sides of an application and passes data between them. Webopedia identifies the following as Middleware software categories:

- TP Monitors
- Distributed Computing Environment (DCEs)
- RPC systems
- ORBs
- Database access systems and MOMs

The Internet 2 Consortium (Internet2) defines the term Middleware as a layer of software, or "glue", but this layer of software is clearly restricted to that between the network and the applications. Here, Middleware software provides a more specific and well-defined set of services such as identification, authentication, authorization, directories, and security in the form of certification. The Network Design Manual (Network Computing, 1995) defines the term by listing the services it supplies to client and server. They describe two sets of services, basic and advanced. The basic functions are:

- Client/server connectivity. Middleware to provide the mechanism by which network applications handle formatting of data for transport and communication across the network.
- Platform transparency. Middleware that hides the differences between platform specific encoding (e.g., between EBCDIC and ACSII).
- Network Transparency and Isolation. Middleware that makes networking choices transparent to application programmers.
- Application and Tool Support. Middleware that is responsible for presenting its own Application Programming Interface (API) to client applications that might use it, and (for most generic Middleware) to the server as well.
- Language Support. Middleware that provides transparency across different SQL dialects.
- RDBMS Support: Middleware that provides a level of transparency across different data storage formats.

Some of the advanced services Middleware supplies are:

- Single System Login. Most advanced Middleware services let the user login in only once to the Middleware's security, which in turn handles authentication across different RDBMSs.
- Enhanced Security. Some Middleware vendors offer better security options like card key solutions, fingerprint readers, and data encryption of sensitive data.
- Location Transparency. Advanced Middleware solutions offer centralized naming services with some level of distribution.
- Application Oriented Services. Different application services like transaction monitoring and message queuing may be provided.
- Database oriented services. Some Middleware may allow multiple RDBMSs to join transparently.
- Management. Some vendors offer products to monitor the health of their Middleware networks.
- Interaction with other network services. Support for other vendors services (security, directory, management) can make Middleware fit into the existing enterprise networking environment much more easily.

David Linthicum (1997, ch. 3) asserts that Middleware gives developers an easy way to get at external resources via a common set of application services such as an API." External resources may include a database server, a queue, a 3270 terminal, or access to real-time information (e.g., a real-time stock market feed). Linthicum goes on further to say that Middleware is "a means of connecting clients to servers without having to negotiate through many operating systems, network, and resource server layers."

Linthicum classifies Middleware products into one of two categories: (1) Primitive Middleware, and (2) Database-oriented Middleware.

Primitive Middleware is that Middleware layer that ties together many different systems to create a single logical system. Linthicum refers to Primitive Middleware as a "virtual system" and calls Primitive Middleware virtual system

Middleware, or "the plumbing and wiring of distributed computing". He classifies that following as Primitive Middleware:

-- remote procedure calls (RPCs)

-- message-oriented Middleware (MOM)

-- object request brokers (ORBs)

-- the distributed computing environment (DCE)

-- TP monitors, and other mainframe access Middleware (such as APPC, APPN, and CPI-C)

He states that these products are all network and platform-independent and that they facilitate communication through a known platform-independent API found on each platform of the client/server system.

Database-oriented Middleware, in contrast, refers to Middleware built specifically for database access. Database-oriented Middleware may use a primitive Middleware layer, such as RPC or messaging, to move information to and from a database. Microsoft's Open Database Connectivity (ODBC) is a good example of a widely used database-oriented Middleware. Others include JDBC, IBI's EDA/SQL, and many other products.

Nathan Muller in his Encyclopedia of Voice and Data Networking defines Middleware as "a broad category of software that facilitates connectivity between applications, databases, or processes-usually in enterprise client/server environments-despite differences in underlying communications protocols, operating systems, platforms, database formats, and other application services". (Muller, 2000, Part III, M - middleware) He sites application programming interfaces (APIs) and remote procedure calls (RPCs) as examples of Middleware.

According to Sybase (DirectConnect, 1999), Middleware's original role was that of data access. Today, however, Sybase asserts that Middleware is expected to provide the following additional functionality

- Simplify access to the multiple databases, applications, platforms, and objects common in today's computing environments

- Integrate existing systems easily with new technology as it emerges
- Improve the scalability, performance and reliability of today's 2-tier client/server applications by making it simple to implement multi-tier client/server systems
- Provide the flexibility to partition applications between different computers and platforms
- Simplify administration of the distributed environment by supporting proprietary as well as industry standard management solutions

They significantly broaden the term when they assert that Middleware must be able to address any-to-any computing relationships. This includes:

- Any client, whether a standard desktop application, a Web object, or even a mobile client
- Any data-desktop, relational, non-relational, or complex
- Any architecture-Internet/Intranet, client/server, multi-tier, mainframe, object-oriented, or message-based
- Any platform-Windows, Windows NT, UNIX, Internet, mid-range, MVS, VSE
- Any location-centralized, distributed, centributed, geographic, or remote

In her book, DCOM Explained, Rosemary Evans defines Middleware as "off the shelf connectivity software, which supports distributed processing" at runtime and software that is "used by developers to build distributed software." (1998, ch. 2) It is the software that binds the parts of a heterogeneous system together. She goes on to say that whereas network software handles the communication between one machine and another, Middleware provides the developer with services that support process-to-process communication across the network. Thus, she says Middleware makes all machines appear like one 'virtual machine'

She classifies Middleware products into two groups:

- Database connectivity products, or products provided by companies such as Intersolv, Information Builders, Shadow, Oracle, and Sybase with their ODBC (Open Database Connectivity) drivers and gateways.)
- Other connectivity products not associated with database connectivity.

These include MOM (Message-Oriented Middleware), ORBs (Object Request Brokers), RPC (Remote Procedure Call), and DTPMs (Distributed Transaction Processing Middleware).

In their book High Performance Client/Server, Chris Loosley and Frank Douglas propose a more limited definition of Middleware. (1997, ch. 15) They define Middleware as 'the provider of services to support client-server computing' and as 'the enabler of interprocess communications by providing an application programming interface (API) that isolates the application programmer from the details of the low-level network communication formats and protocols (FAPs). Under their scheme, some of the services that Middleware provides to support client-to-server communication are as follows:

- Directory services, to assist in routing client requests to the location of their intended servers.
- Security services, to prevent unauthorized clients from connecting to a server.
- Message construction, or marshaling or packing the parameters of the client request into a message to send to the server-and then unmarshaling (unpacking) them at the receiving end.
- Character set translation, to support interoperation between client and server platform that use different character formats. (In OSI terms, this function belongs at layer 6, the presentation layer, but Middleware product functions do not always follow the OSI Reference Model, especially at the upper layers.)

The authors assert that there is little agreement on an organizing framework for Middleware, and choose to name the three major classes of Middleware according to "presentation", "application", and "data" – the three kinds of logic found in a

typical enterprise information-processing application. A brief description of each of these Middleware classes follows:

- *Distributed Presentation Middleware* Distributed presentation Middleware is the combination of a Web browser on the client workstation, the HTTP protocol, and a Web server acting together. In the future, the authors see more sophisticated protocols based on DCOM and CORBA emerging, which will move components or "applets" containing presentation logic to the client, thereby blurring the distinction between this layer and the other two.
- *Database Middleware* The Database Middleware ships the SQL requests over the network to the DBMS and later returns data in the other direction to the application. Fat client applications issue SQL requests to a relational DBMS such as DB2, Informix, Oracle, SQL Server, or Sybase. These SQL requests are handled by proprietary database Middleware supplied by the DBMS vendor. The other most common examples of database Middleware are database gateways, typically supporting an ODBC API, that provide remote database access.
- *Application Middleware* Application Middleware includes the following categories of products :

  -- Point-to-point, or direct messaging Middleware (P2P)

  -- Remote procedure call

  -- MQMs

  -- ORBs

  -- Distributed Transaction Monitors

  Application Middleware, in contrast to the other two, is much more like a general-purpose programming language. It allows two user-written components communicate in any way that suits the application designer and developer. The choice of the desired communication style is the key application design decision. Of the many styles of communication used by processes, the three most popular are:

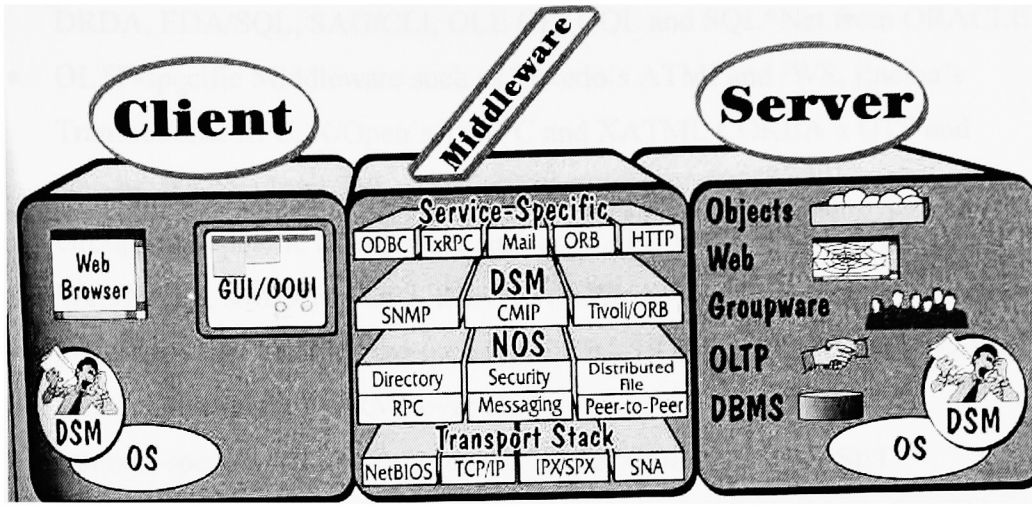
  -- conversations

  -- remote procedure calls

-- object messages

These styles are implemented as interface standards such as CPI-C, the OSF/DCE RPC, and CORBA, respectively. Message queuing introduces an additional style and uses its own API, such as IBM's Message Queuing Interface (MQI). Application Middleware not only connects distributed processes but also can interconnect different types of Middleware.

Orfali and Company (Orfali et al, 1999, p. 44) offer one of the broadest definitions of the term Middleware. They refer to it as the slash ("/") in the term "client/server". Clearly, this goes well beyond the boundaries set by the ICIAR workshop participants. Middleware begins with the client side that is used to invoke a service, and it covers transmission of the request over the network and transmission back of the resulting response. However, Middleware does not include the software that provides the actual service (that falls into the server application's domain), nor does it include the database. On the client side, it does not include the user interface (that falls into the client application's domain). Middleware makes the middle tier in n-tier environments act as a single system, and provides a platform for running server-side components, balancing their loads, managing the integrity of transactions, maintaining high availability and securing the environment. Orfali and Company further propose a new metaphor for describing Middleware in n-tier client/server environments in terms of "pipes" and "platforms". Figure 1 illustrates different services that Orfali et.al., classify as middleware based on their 'pipes' and 'platform' metaphor.



(Orfali et al, 1999, p. 43)

Figure 1: Middleware products/services

Pipes provide the inter-component (and inter-application) communication services, e.g., RPCs, MOMs and ORBs, Secure Socket Layer (SSL) and Lightweight Directory Access Protocol (LDAP). Pipes can be further classified into:

- General pipes provide the substrate for most client/server interactions. They include the communication stacks, distributed directories, authentication services, network time, remote procedure calls, and queuing services. The category also includes network operating system extensions such as distributed file and print services. Products that fall into the general Middleware category include LDAP, X.00, firewalls, and digital certificate servers. SSL, DCE, MS-RPC, Named Pipes, TCP/IP, APPC, IPX/SPX, and NetBIOS. Also included in this category are the MOM products offered by such companies as IBM, BEA, TIBCO, Microsoft, PeerLogic, and Neon Systems.

- Service-specific Pipes accomplish a particular client/server type of service. They include database-specific Middleware such as ODBC, JDBC, SQLJ, DRDA, EDA/SQL, SAG/CLI, OLE DB, OQL and SQL*Net from ORACLE
- OLTP-specific Middleware such as Tuxedo's ATMI and /WS, Encina's Transactional RPC, X/Open's TxRPC and XATMI, CORBA's OTS and Microsoft's DTC and TIP.
- Groupware-specific Middleware such as MAPI, VIM, JavaMail. SMTP, Web/NNTP, s/MIME, POP3, IMAP, Workflow, Lotus Notes.
- Object-specific Middleware such as CORBA/IIOP from OMG, Microsoft's COM+, and Javasoft's RMI-over-IIOP.
- Internet-specific Middleware such as HTTP, CGI, XML, and SET.
- System-management-specific Middleware such as SNMP, CMIP, RMON, DMTF, WFM, JMAPI, WEBEM, and ORBs

Platforms are application servers that run the server-side components. They can be used across multiple operating systems to provide a unified view of the distributed environment, e.g., TP Monitors, Object Transaction Monitors and Web Application Servers.

Newton offers a definition of Middleware that falls completely into the realm of telephony. He asserts that:

> Middleware is software that sits between layers of software to make the layers below and on the side work with each other. Middleware could be almost any software in a layered software stack. In computer telephony, Middleware tends to be software that sits right above that part of the operating system dealing with telephony – TSAPI in Netware and TAPI in Windows – but below the computer telephony application above it that the user sees on their desktop. In short, Middleware is software invisible to the user which takes 2 or more different applications and makes them work seamlessly together.
>
> (Newton, 2001, p. 442)

In a presentation to the Gartner Group's Application Integration Conference in November of 1999, Roy Schulte(1999, p. 3) offered yet another definition of the

term. He defined Middleware as "runtime system software that directly enables application-level interactions among programs in a distributed computing environment." He further elaborated that middleware services generally belong to one of three layers of the OSI model:

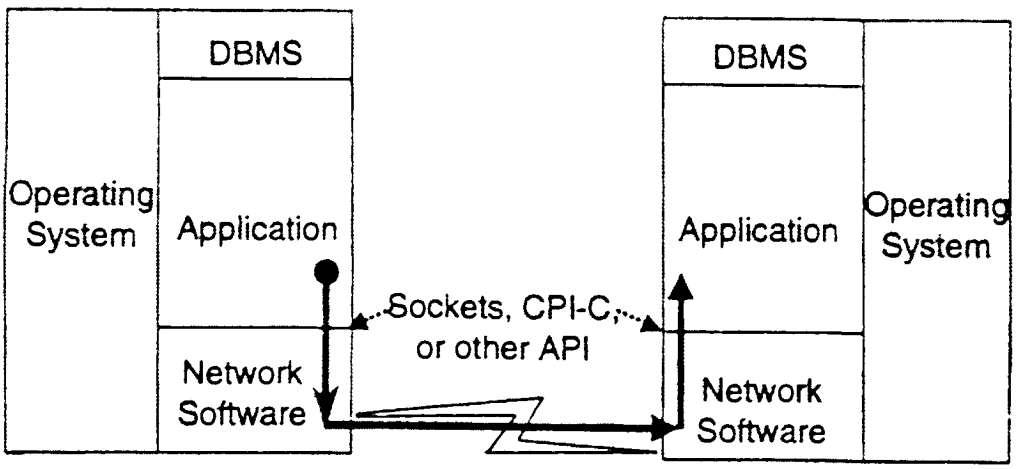
-- the Session layer

-- presentation layer

-- application layer

Middleware provides useful functions like routing, DBMS access, file access, queuing, load balancing and security to name a few. He further adds that implementing a distributed application without middleware is impractical, as it requires the application developer to write to low-level interfaces like TCP-IP sockets and handle many communication-related chores. He illustrates this type of an application with the diagram shown in Figure 2:

Figure 2: Distributed application without middleware



With regards to the taxonomy of middleware, he divides middleware into two layers, Integration Middleware and Basic Middleware. Gartner's middleware taxonomy is illustrated in the diagram in Figure 3 .

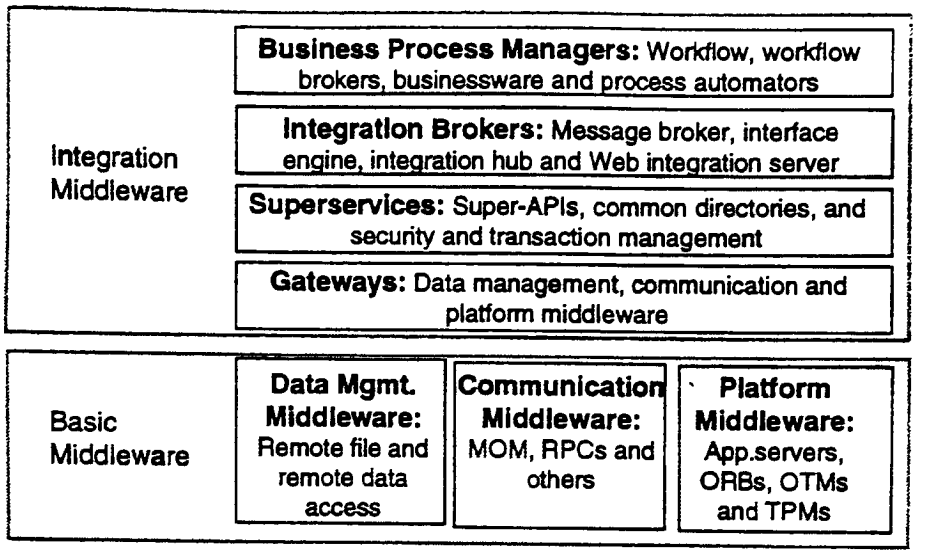


Figure: 3 Gartner's Middleware taxonomy

Basic Middleware is a type of middleware used by a single distributed application. It is made up of the following sub-layers:

- Data Management software, or software related to remote database access, such as ODBC drivers
- Communication Middleware, or software that shields the application developer from having to deal with communication chores like marshalling, routing, addressing, etc. Examples of this type of software are RPCs and MOMs
- Platform Middleware, or a communications software "superset" that insulates programmers from the operating system i.e. It provides memory management and operating system process management services in addition to communication services. Examples include Object Request Brokers and Application Servers

Integration Middleware, is a superset of Basic Middleware. It can help connect independently designed application systems or it may be used in a single or extended application system where programs or DBMSs run on dissimilar platforms. This layer is made up of four sub-layers. From the layer closest to the Basic layer to the farthest, these layers are as follows:

- Gateways, the software that handles communication between dissimilar protocols. It intercepts communications sent and converts it into a form understandable by a dissimilar recipient.
- Superservices which provide some common functionality (eg: directory, transaction management, security, etc) across 2 or more operating systems. They enable a common, super-API to function across disparate underlying technologies
- Integration Brokers, the sublayer that provides transformation (e.g., message translation, both syntactic and semantic) and flow automation (e.g., context-based routing) at the application layer of services
- Business Process Managers, the sublayer that oversees, coordinates and integrates a business process through its business life cycle.

As noted earlier in this paper, the wide variance in these definitions have led several sources to state that the term Middleware is defined either relative to those defining it, ill-defined altogether, or, perhaps so broad in scope as to be useless. Perhaps the clearest indication of this was voiced by the ICIAR Middleware workshop participants, who asserted that the many definitions of the term might be dependent not on what is being defined, but by whom the term is being defined , and on why a definition is being sought. They stated it in this fashion:

> The Workshop participants agreed on the existence of Middleware, but quickly made it clear that the definition of Middleware was dependent on the subjective perspective of those trying to define it. Perhaps it was even dependent on when the question was asked, since the Middleware of yesterday (e.g., Domain Name Service, Public KeyInfrastructure, and Event Services) may become the fundamental network infrastructure of tomorrow. Application environment users and programmers see everything below the API as Middleware. Networking gurus see anything above IP as Middleware. Those working on applications, tools, and mechanisms between these two extremes see it as somewhere

between TCP and the API, with some even further classifying Middleware into application-specific upper Middleware, generic middle Middleware, and resource-specific lower Middleware.

(Aiken et al, 2000, p. 3)

III. General Concepts and Terms

The definitions of Middleware above – and the discussion that follows -- require the use of several concepts and terms used within the Information Technology field. In this section we list and then cite definitions for them.

API

API (application program interface) is a set of routines, protocols, and tools for building software applications; a good API makes it easier to develop a program by providing all the building blocks that programmer uses to assemble an application (webopedia, 'API'). A callable API does not require a pre-compiler to convert SQL statements into code, to compile them, and then bind them to a database. Instead, a callable API allows you to create and execute SQL statements at run time.

Client

A client provides the user interface for the services a system provides. (Orfali et al, 1999, p. 91-95) All client applications request the services from a server. Clients can be classified into three categories: non-GUI, GUI and OOUI. A non-graphical user interface (non-GUI) client generates client requests with a minimum of human interaction (e.g., Automatic Teller Machines, barcode readers, robots and daemon programs). A Graphical User Interface (GUI) is a graphical rendition of a "command line" dialog that is used in dumb terminal/mainframe interaction (e.g., OLTP-type business applications and front-end clients to database servers). The Object-Oriented User Interface (OOUI) is a highly iconic, object-oriented interface that lets you directly manipulate objects on a screen (e.g., decision-support applications and stockbroker workstations).

Data Encryption Standard (DES)

See Encryption

Datagram

A datagram is a packet that includes both source and destination addresses provided by the user, rather than by the network. (Fiebel, 2000, Part I D - datagram) A datagram can also contain data. A message might be sent as multiple datagrams, which may be delivered to the destination in nonconsecutive order. Receipt of a datagram is not acknowledged. Datagram routing takes place at the Network layer of the OSI reference model. Datagram transmission takes place at the Data-Link layer.

Datagram services are provided in connectionless (as opposed to connection-oriented) transmissions. Because connectionless transmissions do not necessarily deliver datagrams in order, datagram services cannot guarantee successful message delivery. Receipt verification is the responsibility of a higher-level protocol, which must be able to assemble the message from the datagrams. Protocols that provide this type of service include UDP (User Datagram Protocol) in the Internet's TCP/IP protocol suite, CLNP (Connectionless Network Protocol) in the OSI reference model, and DDP (Datagram Delivery Protocol) in the AppleTalk protocol suite.

Encryption

"Data encryption is a method of scrambling information to disguise its original content. It is the only practical means of protecting information transmitted over communications networks, but it can also be used to protect stored data. Encryption methods come in two forms: hardware or software. Of the two, hardware-based encryption provides more speed and security." (Muller, 2000, Part III L- LAN Security) Two predominant approaches to electronic encryption are based on shared cipher keys and public cipher keys. With shared private keys, a single key is used to encrypt or decrypt

information. Each pair of users who need to exchange messages must agree on a private key and use it as a cipher to encode and decode their messages. Data Encryption Standard (DES) is based on shared private keys. Public cipher keys use two keys: a public key and a private key. The public key is listed in directories and is available for all to see. Anybody can send an encrypted message encoded with the public key as cipher; the receiver decodes its with the private key. RSA is based on public shared keys.

Message

The message is a delivery mechanism for service requests and replies. (Orfali et al, 1999, p. 15)

Network

A network consists of computers, called nodes or computers which are connected to, or can communicate with, each other in some way. (Fiebel, 2000, N – Network) The nodes run special software for initiating and managing network interactions. With the help of networking software, nodes can share files and resources, and tasks can be distributed over multiple machines on some networks. The following are the main hardware components of a network:

- Nodes: Computers and network interface cards (NICs)
- Topology: Logical and physical
- Connection elements: Cabling, wiring centers, links, and so on
- Auxiliary components: Peripheral devices, safety devices, and tools
- The software components include the following:
- Networking systems: Network operating system (NOS) and workstation software
- Resources, Drivers and various types of server software, e.g., authentication, intranet, and Web servers.
- Tools, Utilities, LAN analyzers, network monitoring software, and configuration managers.
- Applications, Network-aware software

Peer-to-peer communications

Most early client/server applications were implemented using low-level, conversational, peer-to-peer protocols – such as sockets, TLI, CPIC/APPC, NetBios and Named Pipes. (Orfali et al, 1999, p. 159) There were fewer alternatives then. These are hard to code and maintain, so they are losing their popularity. Instead, programmers are now using Remote Procedure Calls, MOMs, and ORBs, which provide higher levels of abstraction.

Pragma

ORBs provide global identifiers – called Repository IDs – to uniquely and globally identify a component and its interface across multi-vendor ORBs and repositories. The repository IDs are generated via pragmas in IDL. The pragma specifies whether to generate them via DCE Universal Unique Identifiers(UUIDs) or via a user-supplied unique prefix appended to IDL-scoped names. (Orfali et al, 1999, p. 483)

Server

A server provides services to clients. Based on the type of services servers provide, a listing of some of the different servers and the services they provide is given below:

File Servers: They are useful for sharing files across a network.

Database Servers: Here the client passes SQL as messages to the server and the server processes the SQL and sends the results back to the client.

Transaction Servers: Here the client invokes remote procedures (or services) that reside on the server with a SQL database engine. These remote procedures on the server execute a group of SQL statements. The network exchange consists of a single request/reply message (as opposed to the database server's approach of one request/reply message for each SQL statement). The SQL statements either succeed or fail as a unit.

Groupware servers: They address the management of semi-structured information such as text, image, mail, bulletin boards, and the flow of work.
Web Application Servers: Basically, a web server returns documents when clients ask for them by name. The client and server communicate using an RPC like protocol called HTTP.
(Orfali et al, 1999, p. 16-20)

Some typical server functions include waiting for client initiated requests, executing numerous requests at the same time, protecting the integrity of shared resources, providing different levels of service priority to its clients, initiate and run background tasks and be robust and scalable. (Orfali et al, 1999, p. 82) Servers may also provide extended services that exploit the distributed potential of networks. Some of these services include supporting different communication stacks, facilities for extending the file and print services over the network, support for Binary Large Objects (BLOBs) and global directories, authentication and authorization services, system management, synchronization of time on networks, database and transaction services, etc.

IV. **Middleware Related Concepts, Protocols, and Software products**

To aid in our analysis, we now list the different concepts, protocols, and software products mentioned in the definitions given in Section 1 of this paper. We present them here with a short description for each.

APPC

See System Network Architecture (SNA)

APPN

See System Network Architecture (SNA)

CGI

CGI (Common Gateway Interface) is the method by which a web server can obtain data from (or send data to) databases, documents, and other programs, and present that data to viewers via the web. (Hamilton, 1999)

CMIP

CMIP short for Common Management Information Protocol, and pronounced see-mip, an OSI standard protocol used with the Common Management Information Services (CMIS). CMIS defines a system of network management information services. CMIP was proposed as a replacement for the less sophisticated Simple Network Management Protocol (SNMP) but has not been widely adopted. CMIP provides improved security and better reporting of unusual network conditions. (webopedia, 'CMIP')

COM+

COM+, or the next generation of COM, was developed to make COM more programmer-friendly and consistent across Microsoft languages. Eventually, COM+ was to provide "a shared common run time that hides most of COM's nastiness from developers – including the IDL, class factories, reference counting, unknown negotiations, class registration and type libraries."
(Orfali et al, 1999, p. 522)

CORBA

CORBA (Common Object Request Broker Architecture) allows intelligent [object] components in different locations on a network to discover each other and then to interoperate with each other over an object bus. (Orfali et al, 1999, p. 474) The CORBA process responsible for this is the Object Request Broker (ORB). It provides global identifiers called Repository IDs that uniquely and globally identify a component and its interface across multi vendor ORBs and repositories. The Repository IDs are system-generated, unique strings used to maintain consistency in the naming conventions used across repositories. An

Interface Definition Language (IDL) is used to specify a component's attributes, the parent classes it inherits from, the exceptions it raises, the typed events it emits, and the methods its interface supports, including the input and output parameters and their data types.

CORBA OTS

See Common Object Request Broker Architecture

CORBA/IIOP

See Common Object Request Broker Architecture

CPI-C:API

See System Network Architecture; See also TP Monitor

DCE

"The Distributed Computing Environment (DCE) from the Open Software Foundation (OSF) creates an open Enterprise/NOS environment that spans multiple architectures, protocols, and operating systems. DCE provides key distributed technologies, including a remote procedure call, a distributed naming service, a time synchronization service, a distributed file system, a network security service, and a threads package." (Orfali et al, 1999, p. 184)

DCE RPC

DCE provides an Interface Definition Language (IDL) and a compiler that facilitates the creation of RPCs. The IDL compiler creates portable C code stubs for both the client and server sides of the application. The stubs are compiled and linked to the RPC run-time library, which is responsible for finding servers in a distributed system, performing the message exchanges, packing and unpacking message parameters and processing any errors that occur.

The most powerful feature of the DCE RPC is that it can be integrated with the DCE security and naming services. (Orfali et al, 1999, p. 185)

Data warehouse

Richard Hackathorn defines a warehouse as "a collection of data objects that have been inventoried for distribution to a business community". (Orfali et al, 1999, p. 271) A warehouse is an active intelligent store of data that can manage and aggregate information from many sources, distribute it where needed, and activate business policies.

Digital certificates

A digital certificate is an electronic file that serves as a container for a public key. (Orfali et al, 1999, p. 150-151) It serves as a user's electronic ID. As an example, an X.500 certificate contains a variety of information, including the owner's public key the owner's name, some attributes associated with the owner, the name of the encryption algorithm, the issuer of the certificate and an expiration date. The Secure Electronic Transactions (SET) protocol uses digital certificates. It automates all the steps that are currently associated with the processing of a credit card in a store – including the phone calls and transaction slips. SET uses digital certificates to authenticate every player.

Distributed Time Services

Network Operating Systems use Distributed Time Services to periodically synchronize the clocks on every machine in the network. (Orfali et al, 1999, p. 139) The Network Operating System typically has an agent on each machine (DCE calls it a "Time Clerk") that asks the timeserver for the correct time and adjusts the local time accordingly. Typically, the universal time to which all machines must be synchronized is the Universal Coordinated Time (UTC).

Distributed Management Task Force (DMTF)

The Distributed Management Task Force (DMTF) is an industry consortium that defines standards to manage all components on a PC, Mac or workstation – including hardware, Operating Systems, applications, storage, and peripherals. The DMTF consortium consists of about 100 formal members. These include the SCO, Symantec, and Sun (Orfali et al, 1999, p. 690)

DRDA

DRDA (The Distributed Relational Database Architecture) from IBM provides an interoperability standard for fully distributed heterogeneous relational database environments. (Orfali et al, 1999, p. 257) To do that, DRDA defines the protocols for database client-to-server and server-to-server interactions. In DRDA terminology, a client is called an Application Requester (AR) and a database server is an Application Server (AS). The AR-to-AS protocol is used for the typical client/server interactions.

DTC (Microsoft)

Distributed Transaction Coordinator (DTC) was introduced in 1997 with SQL Server 6.5. (Orfali et al, 1999, p. 311) It used OLE TP to coordinate transactions, which spanned across more than one SQL Server. Microsoft Transaction Server(MTS) also uses the DTC to manage transactions distributed over different SQL servers. (Orfali et al, 1999, p. 523)

EDA/SQL

Enterprise Data Access/SQL (EDA/SQL), from Information Builders Incorporated(IBI), is a family of open gateway products that uses SQL to access over 72 relational and non-relational database servers. (Orfali et al, 1999, p. 253)

Enterprise Java beans (EJB)

An Enterprise Java bean (EJB) defines a packaging mechanism for server–side components based on JAVA Archive File (JARs), manifests (which maintain entry names and their associated attributes), and deployment descriptors (which contain most of the information about the bean, to include the identification, security roles, transaction demarcation, and any optional environment definition). The container un-JARs the EJB and then runs it based on the instructions it gets from the manifest and the deployment descriptors. (Jguru, 2002)

ESQL

(SQL-92 Embedded SQL) See SQL

External Data Representation (XDR)

See Remote Procedure Call

Firewall

As more computers become networked, often providing access to the public Internet, the possibility of break-in attempts increases. Corporate networks today contain valuable resources that provide access to services, software, and databases. By occupying a strategic position on the network, firewalls work to block certain types of traffic and allow the rest of the traffic to pass. They can also reconfigure themselves to stop successive break-in attempts and provide tools that can be used to track down the source of the attacks.

Types of Firewalls

There are several types of firewalls:

-- packet filters

-- stateful packet inspectors

-- circuit-level gateways

-- application gateways

Some firewall products combine all four into one firewall server, offering organizations more flexibility in meeting their security needs.

Global Directory Services (GDS)

A Global Directory Service (GDS) tracks all of the Network Operating System's resources. Ideally, a distributed directory should provide a single image that can be used by all network applications, including e-mail, system management, network inventory, file services, Remote Procedure Calls, distributed objects, databases, authentication and security. The directory server is implemented as a distributed, replicated, object database, with Netware Directory Service and Lightweight Directory Access Protocol. Some of the standard APIs to interface to these directories are as follows: proprietary directory specific APIs and class libraries, LDAP and X.500 APIs and java classes, distributed object interfaces and meta-directory services and scripts.

GUI

(Graphical User Interface) See Client

Groupware

Groupware is the software that supports the creation, flow, and tracking of non-structured information in direct support of collaborative group activity. (Orfali et a, 1999, p. 388) Groupware deals with highly unstructured data (i.e., text, image, graphics, faxes, mail and bulletin boards) captured as a document object. It provides the tools to capture data at the point of entry, organize it into a document, and help end users create document databases. It also provides support for querying, managing, distributing and navigating through document databases.

HTTP

Hypertext Transfer Protocol (HTTP) is used to access resources that live in URL space. (Orfali et al, 1999, p. 572) HTTP is a stateless RPC that 1) establishes a client/server connection, 2) transmits and receives parameters including a returned file, and 3) breaks the client/server connection. HTTP

clients and servers use the Internet's MIME data representations to describe (and negotiate) the contents of messages.

IBM EDA/SQL

Enterprise Data Access/SQL (EDA/SQL), from Information Builders Incorporated (IBI), is a family of open gateway products that uses SQL to access over 72 relational and non-relational database servers-an industry record. (Orfali et al, 1999, p. 253-255) There are many key EDA/SQL components. API/SQL is another "common" Call-Level Interface (CLI) that uses SQL-92 as the standard database access language. API/SQL will pass through SQL calls that it does not recognize. These calls can be passed through asynchronously. EDA/Extenders are utilities that allow API/SQL calls to be issued from within existing products that support some form of dynamic SQL. You can think of extenders as "redirectors" of SQL calls. EDA/Link supports over 12 communication protocols including Net BIOS, Named Pipes, SNA, TCP/IP, and DECNet. EDA/Link provides password verification and authentication; it also handles message format translations. EDA/Server is a multithreaded catcher that typically resides on the target database server machine. It receives client requests and translates them into server specific commands. EDA/Data Drivers provide access to data in over 72 different formats. These drivers take care of any variations in syntax, schema, data types, catalog naming conventions, and data representation. EDA/SQL is used for decision support systems and data warehousing. However EDA/SQL does not provide the robust transactional support needed for OLTP and production type database access.

Interface Definition Language (IDL)

See both CORBA and Remote Procedure Call

IIOP

Internet Inter-Orb Protocol (IIOP) is a protocol developed to make it possible to request data objects over distributed systems—specifically, over the Internet. [Fiebel, 2000, P – Protocol, IIOP] The protocol provides a way for clients to find, request, and use data objects when necessary. Such requests are made through an ORB (object request broker), which mediates between clients and a back-end database application or a server managing access to such a database. IIOP is used in CORBA (common object request broker architecture), which is a platform-independent specification that allows clients to interact across distributed networks—both with each other and with objects that are located in remote locations. IIOP is based on GIOP (General Inter-ORB Protocol), but has been optimized for use on the Internet. Microsoft's protocols for COM+ (Component Object model) and DCOM (Distributed COM) represent a different way to deal with distributed objects.

IMAP; IMAP4

See Internet Protocols

Internet Mail Protocols

Internet Mail Protocols are a collection of standards and recommendations codified by the Internet Engineering Taskforce (IETF) for standardizing internet mail. (Orfali et al, 1999, p. 411-412) The key mail related Internet protocols are POP3, SMTP, SMTP/E, IMAP4, MIME, S/MIME, and LDAP. Post Office Protocol (POP) and its current version 3 (POP3) allows a client to send and receive e-mail to and from mail servers. The client logs on to the server and sends outgoing mail; the server sends incoming mail to the client and deletes it from its store. The server sends the outgoing mail via the Simple Mail Transfer protocol (SMTP) backbone, which defines how messages are routed between e-mail servers. Multipurpose Internet Mail (MIME) defines the mail message's data types. SMTP only defines the ASCII text; MIME

defines non-text attachments and data types such as image, video and audio. Extended Simple Mail Transfer protocol (SMTP/E) defined in RFC 1651 adds some of the sophisticated messaging features that are common in the proprietary protocols, to include priorities, read receipts, message size negotiation, and error tracing. Internet Message Access Protocol (IMAP) and its current Version 4 (IMAP4) allow a client to access and manipulate e-mail on a server. The client can remotely view the message subject lines and then selectively download incoming messages. IMAP4 supports shared mail folders and folder hierarchies, while POP does not. Eventually IMAP will become the primary client/server protocol for e-mail, it will replace POP as well as some of the proprietary e-mail protocols. Secure Multipurpose Internet Mail (S/MIME) is a secure version of MIME. S/MIME lets you sign an e-mail and also send it in a sealed envelope. S/MIME signs the message by making a message digest of the content and then encrypting the digest using the sender's private key. To seal an envelope, S/MIME encrypts all the contents using the triple DES secret key algorithm combined with a public key scheme.

IDL

See Corba

Internet Key Exchange (IKE)

See IPSec

IPSec

IP Security (IPSec) is the IETF's new IPv6 security protocol. It is a wire level security protocol used on the web (Orfali et al, 1999, p. 196) It provides authentication and encryption at the OSI Layer 3 (the IP layer). In contrast, SSL operates at OSI Layer 4. Consequently, IPSec provides an even less obtrusive channel security channel than SSL. IPSec requires the Internet Key Exchange (IKE) protocol to securely manage and exchange cryptographic keys between the two ends of a connection. The IKE protocol exchanges keys,

while IPSec encrypts and signs packets. Currently, IPSec does not mandate a specific encryption algorithm because of US export restrictions. However, most IPSec implementations currently support DES and Triple DES for encryption, and MD5 or SHA-1 for non-tampering and digital signatures.

IPX/SPX

In Novell's NetWare, IPX and SPX are the network protocols responsible for ensuring successful internetwork communications.

IPX

Internet Packet Exchange (IPX) is a Network layer protocol, and it is responsible for addressing and routing packets to nodes on other networks. IPX assigns and works with Network layer addresses, as opposed to Physical layer addresses, which are assigned by the network interface card (NIC) manufacturers. The IPX protocol uses the services of the Data-Link layer, and it provides services to the SPX (Sequenced Packet Exchange) protocol in the next higher layer. The IPX protocol is a connectionless protocol. This means that it doesn't need a fixed connection between source and destination. The protocol can send different packets along different routes, and doesn't need to worry about the sequencing. IPX is also a datagram protocol. This means that each packet comes with everything you wanted to know about it. With this information, a higher level protocol at the receiving end can reassemble the packets in sequence.

SPX

NetWare's Transport layer Sequenced Packet Exchange (SPX) protocol provides a connection-oriented link between nodes.(Fiebel, 2000, P – Protocol, IPX, SPX) A connection-oriented protocol is one that first establishes a connection between sender and receiver, then transmits the data, and finally breaks the connection. All packets in the transmission are sent in

order, and all take the same path. This is in contrast to a connectionless service, in which packets may use different paths.

The SPX protocol ensures that packets arrive at their destination with enough sequence information to reconstruct the message at the receiving end, and also to maintain a connection at a specified level of quality. To accomplish this, SPX is responsible for flow control, packet acknowledgment, and similar activities.

JavaMail

A Java API for sending and receiving e-mail.

JDBC

Like the Open Database Connectivity (ODBC) standard and the ISO's SQL/CLI, the Java Database Connectivity (JDBC) standard provides two major sets of interfaces: 1) an application interface that lets you access SQL services in a DBMS-independent manner and 2) a driver interface that DBMS vendors must adapt to their particular databases. (Orfali et al, 1999, p. 245) Like the other CLIs, JDBC uses a driver manager to automatically load the right JDBC driver to talk to a given database.

Jflow

Jflow is a CORBA workflow standard, developed at the urging of the Workflow Management Coalition (WfMC), who were charged with providing the CORBA Interface Definition Language (IDL) bindings for all its interfaces. (Orfali et al, 1999, p. 406) At the urging of WfMC members, the standard was ratified in September 1998. The CORBA standard defines how distributed objects can participate in workflows using ORB based invocations. The CORBA standard defines an object based workflow framework that consists of about 13 interfaces; it covers the WfMC standards for client invocations and workflow interoperability across servers.

JMAPI

Java™ Management API (JMAPI) is a collection of java programming language classes and interfaces that allow developers to build system, network, and service management applications.

LDAP

See Internet Protocols

LotusNotes

Lotus Notes—or, simply, Notes—is a distributed client-server database application for the Windows 3.x, 9x, and NT environments. (Fiebel, 2000, L – Lotus Notes) The Notes database is built around documents, document groupings, and representations of document content. Documents can be form-oriented or unstructured.

Unlike standard database programs, Notes allows users to view the information in individualized ways, and to expand the database in just about any direction or manner desired. Thus, document contents can be organized and made available in different ways to different users or groups. Thus, Notes can be used for messaging, collaborative work, or even online instruction.

In addition to its main functions, Notes provides other auxiliary features—for example, electronic mail, a calendar, and scheduling capabilities.

MAPI

MAPI (Messaging Application Programming Interface) is an interface for messaging and mail services. Microsoft's MAPI provides functions for using Microsoft Mail within a Microsoft Windows application. (Fiebel, 2000, M - MAPI) Simple MAPI consists of 12 functions, such as MapiDelete-mail(), MapiReadMail(), and MapiSendMail(). By calling these functions in the

appropriate manner and combination, a Windows application can address, send, and receive mail messages while running.

Message Oriented Middleware (MOM)

MOM allows general-purpose messages to be exchanged in a client/server system using message queues. (Orfali et al, 1999, p.170-171) MOM is a key piece of many client/server products. It is also used to create nomadic client/server systems that can accumulate outgoing transactions in queues and do a bulk upload when a connection can be established with an office server. MOM hides from the applications the details of the application-to-server communications and typically provides a simple high-level API to its services. Communication is accomplished by placing messages onto a queue at one end, and by taking messages off at the other. The server also sends back the reply via a message queue. MOM products provide their own NOS services, to include hierarchical naming, security. It provides a layer that isolates applications from the network. The server instances can take messages off the queue either on a first-in/first-out basis or according to some priority or load-balancing scheme. In all cases, a message queue can be concurrently accessed. The servers can also use messaging filters to throw away the messages they don't want to process, or they can pass them on to other servers.

MIME

See Internet Protocols

Named Pipe

A Named Pipe provides highly reliable, two-way communication between a client and a server. (Orfali et al, 1999, p. 164) It provides a "file-like" programming API that abstracts a session-based two-way exchange of data. Using Named Pipes, processes can exchange data as if they were writing to, or reading from a sequential file. Named Pipes run on Net BIOS, IPX/SPX, and TCP/IP stacks.

Net BIOS

NetBIOS is an interface and an upper-level protocol developed by IBM for use with a proprietary adapter for its PC Network product. (Fiebel, 2000, P – Protocol, NetBIOS) NetBIOS provides a standard interface to the lower networking layers. The protocol's functionality actually ranges over the top three layers (session, presentation, and application) in the OSI reference model. Essentially, the protocol provides higher-layer programs with access to the network. NetBIOS can also serve as an API (Application Program Interface) for data exchange. As such, it provides programmers with access to resources for establishing a connection between two machines or between two applications on the same machine.

NetBIOS provides four types of services:

- Naming for creating and checking group and individual names, and for deleting individual names.
- Datagram support for connectionless transmissions that make a best effort to deliver packets, but that do not guarantee successful delivery.
- Session support for transmissions in which a temporary virtual circuit is established for the duration of a session so that delivery of packets can be monitored and verified.
- General services for resetting adapter states, canceling application commands when possible, and so on.

NETBEUI

NetBIOS Extended User Interface(NetBEUI) offers powerful datagram and connection-oriented services. It also offers a dynamic naming service based on discovery protocols. Its main weakness is the lack of a network layer. Its other weakness is the lack of security. (Orfali et al. 1999, p. 164)

NNTP

Network News Transport Protocol (NNTP) provides the ability to participate in discussion groups across the Internet. (Fiebel, 2000, P – Protocol, NetBIOS) It is defined in defined in RFC 977. [In the Internet community, a series of documents that contain protocol and model descriptions, experimental results, and reviews.] All Internet standard protocols are written up as RFCs.

NOS

Network Operating System (NOS) software is expected to provide the following the following transparencies:

Location transparency

Namespace transparency (able to use the same namespace conventions to locate any resource on the network)

Logon transparency (single password authentication)

replication transparency (unable to tell the number of copies of a resource that exist)

Local/remote transparency

Distributed time transparency (unable to see time differences across servers)

Failure transparency

Administration transparency

New Workflow Systems

New workflow packages address both structured and unstructured process automation needs, integrate with other applications, support drag and drop iconic manipulations for creating workflows and defining business rules, provide tools for designing forms (or importing them from a GUI builder), designing sequential or conditional routes, and scripting languages to specify the business logic. (Orfali et al, 1999, p. 395-397) They also integrate with e-mail, MOM, ORB, publish-and-subscribe, and RPC and provide facilities for tracking work-in-progress, provide users with tools to complete an action,

provide APIs that let developers customize workflow services and provide off the shelf component suites for assembling workflow ODBC.

non-GUI

Non Graphical User Interface See Client

Object Bus

See Common Object Request Broker Architecture

Object Oriented User Interface (OOUI)

See Client

OLE DB

Object Link Embedding for databases (OLE DB) defines a set of interfaces (and classes) that mediate between data providers and data consumers. Data providers are OLE DB components that represent a specific data source - SQL data, spreadsheets, file systems, directories, multi-dimensional data, etc. Providers return data using an OLE DB abstraction called a rowset; a table that can also contain embedded tables in its columns. Data consumers are applications that can access this data. OLE DB provides a set of classes that you can use to consume data. (Orfali et al, 1999, p. 250) Call-Level Interfaces (CLIs) have also been implemented for object interfaces. The two most popular object CLIs are JDBC and OLE DB's ActiveX data Objects (ADO). Both provide a CLI that you can access via object interfaces (and classes) instead of procedural APIs. In addition, they both support distributed data access via off the shelf ORBs.

OLTP

OLTP (Online Transaction Processing) refers to business activities that are carried out electronically. (Fiebel, 2000, O – OLTP) Examples of OLTP include making withdrawals or deposits at an ATM (automatic teller

machine), registering for courses electronically, or purchasing something over the Internet. Special safeguards must be built into such transactions to ensure that there are no effects if something goes wrong and the transaction is not completed. For example, someone trying to get cash from their account through an ATM should not have the requested amount deducted from their account balance if the machine is unable to actually provide the cash. The integrity of transactions is (virtually) ensured by adherence to the ACID (atomicity, consistency, isolation, and durability) principles, which have been formulated for just this purpose.

OQL

The Object Database Management Group(ODMG) – a consortium that includes most of the major ODBMS vendors – defined the Object Query Language(OQL). It is a SQL like declarative language for querying and updating database objects. It supports most commonly used SQL SELECT structures; though it does not support SQL INSERT, UPDATE, or DELETE(it uses java, C++ or Smalltalk extensions for this). (Orfali et al, 1999, p. 535)

Open DataBase Connectivity (ODBC)

The Open DataBase Connectivity (ODBC) is a Microsoft protocol that originally was targeted for database access under Windows; recent ODBC versions are now cross-platform. (Orfali et al, 1999, p. 240-242) ODBC 2.0 defines about 61 API calls that fall into three conformance levels. The first conformance level, the Core level, provides 23 base calls that let you connect to a database, execute SQL statements, fetch results, commit and rollback transactions, handle exceptions, and terminate the connection. The next conformance level, Level I, provides an additional 19 calls that let you retrieve information from the database catalog, fetch large objects (BLOBs), and deal with driver-specific functions. The third conformance level, Level 2, embodies an additional 19 calls that let you retrieve data using cursors; it supports both forward and backward scrolling.

ORB

See Common Object Request Broker Architecture

OTM

An Object Transaction Monitor (OTM) manages a set of containers that manage the server-side components. (Orfali et al, 1999, p. 465-466) You declaratively define and administer the properties of your server-side components by setting their attributes. The container then provides the callback objects that implement the required functionality (or quality of service). The user is responsible for writing the business logic of the application. At run time, the OTM intercepts the incoming calls, invokes the appropriate callback objects within the container, and then passes the request to your object. An OTM provides a framework, or organized environment, for running server-side components. It also calls the system services on behalf of your components based on the attributes you set. It pre-starts pools of objects, distributes their loads, provides fault tolerance, and coordinates multi-component transactions.

OTS

See Common Object Request Broker Architecture

Packet Filter

A packet filter is a mechanism that provides a basic level of network security at the IP level. (Orfali et al, 1999, p. 197-198) Packet filters are typically implemented in routers. Packet filters drop, reject or permit packets based on destination IP address, source IP address and application port numbers.

POP; POP3

See Internet Protocols

Proxy Firewall

Proxy Firewalls, or application firewalls, run a small number of programs called proxies that can be secured and trusted. (Orfali et al, 1999, p. 198) All incoming Internet traffic is funneled to the appropriate proxy gateway for mail, HTTP, FTP, Gopher, IIOP and so on. The proxies then transfer the incoming information to the internal network, based on access of the individual users.

Publish and Subscribe Products

Publish-and-subscribe products broker the distribution of events; they mediate between the producers and consumers. (Orfali et al, 1999, p. 175) The publisher produces events and subscribers consume the events. The software that can do something about this event is then notified. Of course, the software must first register its interest in the event by subscribing to the event or event type. An event broker establishes a channel between subscribers and publishers of events. Producers and consumers of events do not talk directly; they communicate through their broker. This level of indirection makes it possible for a broker to provide value-added services such as event filters, event logs, event queues, event-driven rules, priority event routing, subject based event routing, event multicasting, and event load-balancing.

Remote Procedure Call (RPC)

The Remote Procedure Call (RPC) hides the intricacies of the network from the application by using the ordinary procedure call mechanism. (Orfali et al, 1999, p. 166) A client process calls a function on a remote server and suspends itself until the results are returned to it from the server. The RPC run-time software collects values for the parameters, forms a message, and sends it to the remote server. The server receives the request, unpacks the parameters, calls the procedure, and sends the reply back to the client. An entire environment is needed to start and stop servers, prioritize requests, perform security checks, and provide some form of load balancing. The better

NOSs provide an Interface Definition Language (IDL) for describing the functions and parameters that a server exports to its clients. An IDL compiler takes these descriptions and produces source code stubs (and Header files) for both the client and server. These stubs can then be linked with the client and server code. The client stub packages the parameters in an RPC packet, converts the data, calls the RPC run-time library, and waits for the server's reply. On the server side, the server stub unpacks the parameters, calls the remote procedure, packages the results, and sends the reply to the client. RPCs provide some level of data format translation across systems. Sun RPC requires that clients convert their data into a neutral canonical format using the External Data Representation (XDR) APIs.

RMI-over-IIOP

Remote Method Invocation (RMI) over IIOP is part of the Java™ 2 Platform, Standard Edition, v 1.3. RMI over IIOP provides the ability to write CORBA applications for the Java platform without learning CORBA Interface Definition Language (IDL). (Java RMI-IIOP Documentation, n.d.)

RMON

Remote Network Monitoring (RMON) — now in its second incarnation as RMON-2— is a proposed standard for monitoring and reporting network activity using remote monitors. RMON is designed to supplement the management information obtained and used by the SNMP (Simple Network Management Protocol). (Fiebel, 2000, R – RMON) The original RMON specification called for reporting about fairly low-level activity (packet and error rates, utilization percentages, and so forth) on network segments. RMON-2 was released in early 1997, and it includes provisions for taking a higher-level look at end-to-end performance as well. In particular, RMON-2 provides functions for getting information about the operation and performance of entire networks or of subnetworks in an internetwork. Remote monitors are expected to do their work in a way that is minimally disruptive to

network activity and that makes minimal demands on the available resources. Much of the information that remote monitors provide is summary information, some of which can be obtained passively (by counting packets, error signals, and so on).

RSA

The Rivesi, Shamir and Adleman (RSA) Algorithm is a patented public-key encryption algorithm (named for its inventors). (Fiebel, 2000 R – RSA) This algorithm is used for encrypting messages and for creating digital signatures and certificates. The algorithm could not be cracked for many years, but not for as long as expected. Using the processing capabilities of hundreds of computers and the intelligence of hundreds of colleagues, researchers have determined the keys (prime factors of a very large number) used in this encryption scheme.

S-HTTP

Secure Hypertext Transfer Protocol(S-HTTP) adds application-level encryption and security over ordinary socket-based communications. (Orfali et al, 1999, p. 194) The client and server negotiate their security requirements over an HTTP session; they use a MIME-like protocol to encrypt the contents of their messages. S-HTTP provides the following security checks: 1) it authenticates both clients and servers, 2) it checks for server certificate revocations, 3) it supports certificate chaining and certificate hierarchies, 4) it supports digital signatures that attest to a message's authenticity, 5) it allows an application to negotiate the security levels it needs, and 6) it provides secured communications through existing corporate firewalls. Like SSL, S-HTTP incorporates public key cryptography from RSA. In addition, it supports traditional shared-secret (password) and Kerberos-based security systems.

S/MIME

Secure Multipurpose Internet mail Extensions – lets you sign an e-mail and also send it in a sealed envelope. S/MIME signs the message by first making a message digest of the content and then encrypting the digest using the sender's private key. To seal an envelope, S/MIME encrypts all the contents using the triple-DES secret key algorithm combined with a public key scheme.

SAG/CLI

In 1988, 44 database vendors created a consortium called the SQL Access Group (SAG), which was to provide a unified standard for remote database access. They developed one the most successful industry open standards. The SQL/CLI, they developed requires the use of intelligent database drivers that accept a CLI call and translate it into the native database server's access language. The CLI requires a driver for each database to which it connects. (Orfali et al, 1999, p. 239-240)

SMTP

Simple Mail transfer Protocol(SMTP) defines how messages are routed between e-mail servers. SMTP only defines the ASCII text; MIME defines non-text attachments and data types such as image, video and audio. (Orfali et al, 1999, p. 412)

SNA

System Network Architecture (SNA) is an IBM product that has evolved from a basic network infrastructure into a true distributed operating system that supports cross-network directory services, transparent network access to resources, common data streams, and integrated network management. (Orfali et al, 1999, p. 165) Advanced Peer-to-peer network (APPN) is the network infrastructure responsible for this "true distribution". APPN creates an SNA network without the mainframe-centric hierarchy of traditional SNA configurations. APPN allows LU 6.2 SNA applications, using APPC or CPI-C APIs, to take full advantage of peer networks. Common Programming Interface for Communications (CPI-C) builds on top of APPC and masks its complexities and irregularities. Every product that supports APPC has a

slightly different API. CPI-C fixes that problem. The CPI-C API consists of about 40 calls; APPC consists of over 60 calls. Most of these calls deal with configuration and services.

SMTP and SMTP/E

See Internet Protocols

S/MIME

See Internet Protocols

SNTP

Sockets

Sockets were introduced as the Unix BSD 4.2 generic interface that would provide Unix-to-Unix communications over networks. (Orfali et al, 1999, p. 160) In 1985, SunOS introduced NFS and RPC over sockets. In 1986, AT&T introduced the Transport Layer Interface (TLI) which provides functionality similar to sockets but in a more network-independent fashion. Sockets are supported on virtually every operating system. Three most popular socket types are stream, datagram, and raw. Stream and datagram sockets interface to the TCP and UDP protocols, and raw sockets interface to the IP protocol.

SQL

SQL, now an ISO standard, is a powerful, set-oriented language, consisting of few commands that are used to manipulate, define and control data that is stored in a relational database. [Orfali et al, 1999, p. 208]

SQLJ

SQLJ is Java's embedded SQL. SQLJ lets you directly insert SQL statements inside your java programs. (Orfali et al, 1999, p. 237-239) SQL-92 Embedded SQL (ESQL) is an ISO SQL-92 defined standard for embedding SQL

statements "as is" within ordinary programming languages. An SQL Call-Level Interface (SQL-CLI) is a callable SQL API for database access; it is an alternate to Embedded SQL.

SQLJ

See SQL

SQL CLI

(Call-Level Interfaces) See SQL

SSL

Secure Sockets Layer (SSL) is a secured socket connection. (Orfali et al, 1999, p. 192) SSL is now known as the Transport Layer Security (TLS). Today, all commercial browsers and web servers support SSL; it is the standard for authentication and encryption between web browsers and servers. SSL verifies that the content of the message hasn't been altered. SSL implements a security-enhanced version of sockets that provides transaction security at the transport level. SSL provides a secured communications link without involving the applications that invoke it. The SSL protocol provides: 1) private client/server interactions using encryption, 2) server authentication, and 3) reliable client/server exchanges via message integrity checks that detect tampering. The SSL handshake protocol must first be completed before an application can transmit or receive its first byte of data.

System software

System software means software that is positioned between an application-level program and lower-level operating system, data management and networking services.

(Schulte, 1999, p. 3)

TLS

See SSL

TP Monitors

TP Monitors specialize in managing transactions from their point of origin typically on the client – across one or more servers, and then back to the originating client. (Orfali et al, 1999, p. 337-339) When a transaction ends, the TP Monitors ensures that all the systems involved in the transaction are left in a consistent state. In addition, TP Monitors know how to run transactions, route them across systems and load balance their execution, and restart them after failures. One of the great appeals of TP Monitors is that it is the overseer of all aspects of a distributed transaction, regardless of the systems or resource managers used. A TP Monitor can manage resources on a single server or multiple servers, and it can cooperate with other TP Monitors in federated arrangements. A TP Monitor oversees Process Management (starting server processes, funneling work to them, monitoring their execution, and balancing their workloads), Transaction Management (guaranteeing the ACID properties to all programs that run under its protection), and Client/server communications management (which allows clients -- and services -- to invoke an application component in a variety of ways). Transactional client/server exchanges use highly segmented versions of traditional communication models such as RPCs, queues, ORB invocations, publish-and-subscribe, and conversational peer-to-peer. XATMI, TxRPC and CPI-C are some standard transactional communication programming interfaces.

Triple DES

(Triple Data Encryption Standard) See Encryption

TxRPC

See TP Monitor

X/Open's SAG Call-Level Interfaces

The SQL Access Group (SAG) provides a unified standard for remote database access, and requires the use of intelligent database drivers to accept a

CLI call and translate it into the native database server's access language. (Orfali et al, 1999, p. 239-240) The CLI requires a driver for each database to which it connects. Each driver must be written for a specific server using the server's access methods and network transport stack. Currently, the SQL/CLI only supports dynamic SQL and provides functions that correspond to the SQL-92 specification. The API calls allow you to connect to a database through a local driver (3 calls), prepare SQL requests (5 calls), execute the requests (2 calls), retrieve the results (7 calls), terminate the statement (3 calls), and terminate a connection (3 calls) The CLI is just another SQL wrapper – it neither adds nor subtracts from the power of SQL.

XAMTI

See TP Monitors

V. Deriving the core elements of a definition of Middleware

In Sections I, we examined some definitions of middleware obtained from the literature. We then explained the various terms associated with each of them in sections II and III.

To evaluate the consensus among these definitions, we developed a table that lists the major assertions regarding middleware made by each definition (Appendix A). We also listed the different product categories that each definition provides as examples of its definition of Middleware (Appendix B). Using these tables, we attempt to answer each of our original questions.

We assert that each of our questions address a key issue regarding middleware. We will attempt to determine how each definition answers each question. Presumably, if a majority of the definitions we have collected provide the same, or similar answers, to these questions, it will help us create a clearer and more succinct definition of the term.

1. **Should Middleware provide software that provides or supports transparent and "seamless" distributed environments?**

We hold that transparency and seamlessness is a *key* measure of Middleware. Simply stated, transparency or "seamlessness" is a "condition in which an operating system or other service allows the user access to a remote resource through a network without needing to know if the resource is remote or local." (Webopedia, 'transparency') Is this assessment in agreement with the definitions we have collected?

We operationalize our question into determining if each definition asserts something like the following:

> *Middleware serves as a tool to develop, integrate, and support successful distributed applications*

Good distributed applications *by definition* exhibit transparency and seamlessness. Application developers today typically have to develop applications that are distributed across different platforms, that are (1) distributed geographically, (2) access different databases, (3) operate on different platforms and (4) communicate across different protocols. Developers must write to interfaces and handle complex communication chores as well as develop the business logic for organizations relying on such seamless and transparent applications. Is the software that is developed to create such applications deemed to be Middleware?

We established two possible answers to this question:

Yes – Middleware should serve as a tool to develop, integrate, and support transparent and "seamless" distributed applications

No – Middleware does not serve a tool for distributed applications to develop, integrate, and support transparent and "seamless" distributed applications

We analyzed our definitions looking for either an explicit answer to this question, or obvious evidence that would support an inference one way or the other. Our results are shown in Table 1.

Table 1. Middleware's role to multi-process interaction

Alternatives	**Definitions**	**Count of definitions**
Yes	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	13
No		0

How obvious it may appear, the 13 definitions agree that any software that can do the following should be classified as Middleware:

- Provide service that provides transparent application-to-application interaction across the network
- Act as a service provider for distributed applications
- Provide services that are primarily used by distributed applications (e.g., RPCs, ORBs, Directories, name-resolution services, etc.)

It makes sense that a definition of Middleware would embody a requirement that it contains a tool, or tools, and/or an environment in which distributed applications could be fashioned, integrated and supported. With the complex needs of any medium/large scale organization, supporting an entire organization's IT requirements with a single vendor's proprietary product or products seems impractical as well as risky, especially with the IT business evolving so rapidly. Similarly, developing a business specific solution to integrate applications distributed through the organization would require an extremely skilled technical staff and high development cost. Middleware tackles these issues by handling the complex environment and allowing application developers to focus on developing business logic.

Some examples of software that provides interaction between applications as its primary function are RPCs and ORBs. However each definition may use its own words to suggest the above functionality.

The core function of middleware is to allow multiple processes to interact with each other. This function has evolved over time from simple database access services which translated between the different proprietary SQL dialects (used by databases) to more complex services like message queuing, load balancing and remote execution.

2. **At what level does Middleware required to provide transparency?**

Transparency, however, can be achieved at any number of levels within an application. Which levels should Middleware address?

Before we begin, we summarize the following about the idea of transparency, based loosely on that defined for most Network Operating systems:

Platform transparency

Software that allows the same application to be run on multiple platforms/operating systems is said to provide platform transparency.

Network Transparency

Software that allows the same application to be used over multiple communication stacks provides network transparency.

Database transparency

Applications that operate independent of the underlying database(s) exhibit database transparency. This transparency is achieved through database access software like ODBC.

Location Transparency

The location of the applications, i.e., if they are on the same machine, on a client/server or distributed on some machine on the network does not affect the application.

Note that platform, network and location transparency generally go hand in hand with each other.

Operationalizing again, we see that some of the services that help achieve platform, network, and location transparency are:

- Name resolution services
- Directories
- Higher-level application like RPCs, ORBs and MOMs.

Definitions 1 through 13 have slightly differing opinions as to the functions middleware provides in terms of transparency at the various levels. Our judgements are recorded in Table 2 below.

Table 2. Middleware and transparency

Transparency Level	Definitions	Count of definitions
Platform transparency	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13	12
Network transparency	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13	12
Database transparency	5, 6, 7, 8, 9, 10, 11, 13	7
Location transparency	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13	12

We can conclude that most people agree that Middleware is that level of software required to achieve platform, location, and network transparency. However, an interesting point to note is that almost half of the definitions do not include database transparency as something achieved by Middleware. One of the possible reasons for this might be that, for some, ODBC and JDBC is considered software and therefore Middleware, while others consider ODBC and JDBC to be standards and not software per se. Nevertheless, more than half (7 of the 13 definitions) include database transparency as a goal of Middleware products.

3. ***At what layers of the OSI model does (or should) Middleware operate?***

Some of the definitions in Section I provide insight into which layers of the OSI model middleware belongs to. These definitions are numbered from 1 through 13 in the order of their appearance in section I.

Before we begin, we summarize for completeness the activities of each level of the OSI model.

Application Layer

Software at this layer provides support for end-user application services.

Some examples of application level services are:

- Identification
- User-authentication services
- File transfer
- e-mail
- Authorization services and
- Transaction Processing services
- Security through data encryption

Presentation Layer

Software at this layer makes the networking and platform choices transparent to the application developer by translating data from application to network format and vice-versa. Both database access systems and RPCs operate at this level.

Session Layer

The session layer initiates, coordinates, and terminates conversations, exchanges and dialogues between the applications at each end. RPCs, MOMs, ORBs operate at this level as do identification, authentication, security and directory services.
Though security exists at multiple layers of the OSI model, Access Control Lists (ACLs) and digital certificates are examples of session layer security.

Transport Layer

This layer describes the quality and nature of the data delivery. It is responsible for end-to-end error recovery (retransmissions, etc) and flow control. Security could also be a transport level service (SSL and S-HTTP). and can be done even at the network layer. Communication protocols like TCP/IP, NetBIOS and TSAPI fall in the transport layer.

Network Layer

It provides switching and routing technologies, creating logical paths for transmitting data from node to node. Other functions of this layer include forwarding, inter-networking, error handling, congestion control and packet sequencing.

Data Link Layer

Data packets are encoded and decoded into bits. It furnishes addressing, framing and check summing of Ethernet packets.

Physical Layer

This layer defines the physical and electrical properties of various communication media and the interpretation of the exchanged signals.

Each definition was inspected. Where the definition either suggested or explicitly encapsulated the functionality of a given OSI level, it was counted for that level. If the definition neither suggested or explicitly encapsulated the functionality of a given OSI level it was not counted at that level. The results are given in Table 3. Note that though Definition 1 roughly characterizes middleware at the Session and Presentation layers, it also states that certain end-user support applications like accounting and billing belong to a second level of middleware. For this reason we have listed definition 1 under the Application layer as well. Also, Definition 8 discusses the ability of middleware to address multiple data desktops or databases. This is achieved through various data access services located at the Application layer.

Table 3. Middleware and OSI layers

OSI Layers	Definitions	Count of definitions
Application Layer	1, 2, 3, 5, 6, 8, 9, 10, 11, 13	10
Presentation Layer	1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13	11
Session Layer	1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13	12
Transport	10, 11	2

Layer		
Network Layer	10, 11	2
Data Link Layer	10	1
Physical Layer		0

The layers at which middleware operates is a key issue discussed directly or indirectly by all these definitions. Interestingly, most definitions suggest that the definition of Middleware should be confined to the *Session layer of the OSI model and higher*. A minimum of ten definitions put Middleware at the Session, Presentation and Application Layers. Less than 10% of the definitions suggest that middleware services operate below the Session layer implying that middleware performs data translation services between platforms and networks as well as communication services but middleware services are built above the communication protocol, routing and packet creation/transmission layers. In general, then, the functionality at levels 1 through 4 are *not* considered to be the purview of Middleware. This includes such services as SSL, S-HTTP, TCP/IP, NetBIOS and TSAPI.

4. **What services must be supplied by Middleware?**

Middleware must provide software services to achieve transparency and seamlessness in distributed applications as well as provide network, database, platform and location transparency. We now ask ourselves what kinds of software services are *most core* to a definition of Middleware.

We examined the definitions to determine the different services and software objects germane to Middleware. Software services and software objects that were cited by more than eight definitions are listed here. In order by count, they are:

- Authentication
- Security
- Authorization
- Identification
- APIs

The purpose for each of these functionalities and/or objects is briefly outlined below.

Security

Security is broadly defined and can be provided at different layers of the OSI model (discussed in question 3). As used in the current context, security services are in the form of data encryption and digital certificates.

Directory, Authentication, Authorization, Identification

In distributed systems, these services (described in detail in section III) provide the basis for the secure invocation of remote applications.

All of these services are symbiotic in nature and rarely exist in isolation. Hence, definitions that include one of these are assumed to imply that the other three provide the same service, unless it is explicitly stated otherwise.

Directories

Through directories, application developers can discover information about network resources, network status, remote services, etc.

API

In the case of middleware, APIs support the application developers' interface to the services middleware provides. A good API is standardized and well documented.

The tabulation is given in Table 4.

Table 4. Required Middleware software services and software objects

Services	Definitions	Count of definitions

Authentication	1, 2, 3, 4, 5, 6, 9, 10, 11,13	10
Security	1, 2, 3, 4, 5, 6, 9, 10, 11, 13	10
Authorization	1, 2, 3, 4, 6, 9, 10, 11, 13	9
Identification	1, 2, 3, 4, 6, 9, 10, 11, 13	9
Directories	1, 2, 3, 4, 6, 9, 10, 11, 13	9
API	2, 3, 5, 6, 9, 10, 11, 13	8

Assuming that the number of times a service is mentioned, the majority of the definitions rank services associated with legal access to an application as core to Middleware, along with valid, standardized APIs for application development.

It should be noted that there are numerous higher-level software implementations that provide other wrappers to the above-mentioned services. Some examples of these higher-level services (discussed earlier in this paper) are:

- TPMs
- RPCs
- ORBs

We assert by inspection, however, that core to any such wrappers are the services mentioned above.

It is clear from the above table that though there is remarkable consensus among the definitions about the goals of middleware, there is less consensus regarding a base set of services middleware provides to achieve its objectives. Definitions 7, 8 and 12 do not discuss this question at all. There does seem to be a stronger consensus among the subset of definitions not including 7, 8 and 12. Acknowledging the evolving nature of middleware technology, we expect these core services to change in accordance with the changing environment and to be in a greater state of flux than the primary goals of middleware.

VI. **Conclusion**

Middleware is software primarily used in distributed environments to provide multi-process interaction. Analysis suggests that the term "Middleware"

should be confined to software operating at the Session, Presentation and Application layer of the OSI Model. It is clear that Middleware hides the complexity inherent in distributed environments by providing platform, network, and location transparency. Database transparency may be a goal of Middleware, but we surmise that the idea that ODBC and JDBC may be treated more as a standard and therefore not qualify as Middleware "software" The majority of the definitions rank services associated with legal access to an application as core to Middleware, along with valid, standardized APIs for application development, as core to the definition of middleware.

Appendix A. Major Assertio

md1	md2	
OSI, services	connectivity software	softw conne separ appli
above Transport layer	set of enabling services	separ that s glue l two a
below application layer	allow multiple processes on multiple machines to interact on a network	

Missing Page

Missing Page

Missing Page

Missing Page

REFERENCES

1. Aiken, B., Strassner, J., Carpenter, B., Foster, I., Lynch, C., Mambretti, J., Moore, R. & Teitelbaum, B.(2000), Network Policy and Services – A report of a workshop on Middleware. Retrieved April 12, 2002 from http://www.ietf.org/rfc/rfc2768.txt

2. Altman, R. & Schulte, R.(2000), Application Integration Middleware Market [Electronic version]. Retrieved April 12, 2002 from http://www3.gartner.com/DisplayDocument?id=309233&acsFlg=accessCanBuy

3. Bray, M.(1997), Middleware- Software Technology Review. Retrieved April 12, 2002 from http://www.sei.cmu.edu/str/descriptions/middleware.html

4. DirectConnect (White Paper business)(1999). Retrieved April 12, 2002 from http://my.sybase.com/detail?id=204837

5. Fiebel, W.(2000), The Network Press Encyclopedia of Networking [Electronic Version], 3rd ed, USA: Sybex. Retrieved April 12, 2002 from Books 24x7 database.

6. Hamilton, J. (1999), CGI Programming 101. Retrieved April 12, 2002 from http://www.cgi101.com/class/

7. Core Middleware,(n.d.). Retrieved April 12, 2002 from http://Middleware.internet2.edu/core/

8. Java RMI-IIOP Documentation(n.d.), Retrieved April 12, 2002 from http://java.sun.com/j2se/1.3/docs/guide/rmi-iiop

9. Jguru(2002), Enterprise JavaBeans Technology Fundamentals Short Course. Retrieved April 12, 2002 from http://developer.java.sun.com/developer/onlineTraining/EJBIntro/EJBIntro.html

10. Linthicum, D.(1997), David Linthicum's Guide to Client/Server and Intranet Development [Electronic Version]. Retrieved April 12, 2002 from Books 24x7 database.

11. Loosley, C. & Douglas, F.(1997), High Performance Client/Server [Electronic version] Retrieved April 12, 2002 from Books 24x7 database.

12. Muller, N.(2000), Desktop Encyclopedia of Voice and Data Networking [Electronic Version]. 1. Retrieved April 12, 2002 from Books 24x7 database.

13. Network Computing INDM Middleware Guide TOC [Electronic Version], Network Computing(November 25, 1995). Retrieved April 12, 2002 from http://www.nwc.com/shared/printArticle?article=nc/netdesign/cdmwdef.htm&pub=nwc

14. Newton, H. & Horak, R.(2001), Newton's Telecom Dictionary. 17th Ed. USA: CMP Books.

15. Orfali, R., Harkey, D. & Edwards, J.(1999). The Client/Server Survival Guide. USA: John Wiley & Sons.

16. Rock-Evans, R.(1998), DCOM Explained [Electronic Version]. Retrieved April 12, 2002 from Books 24x7 database.

17. Schulte, R.(1999), Middleware Technology Overview, Gartner Group Application Integration Conference.

18. Middleware.(November 9, 2001) Retrieved April 12, 2002 from http://www.webopedia.com/TERM/A/API.html

19. Webopedia, 'CMIP'(October 31, 2001) Retrieved April 12, 2002 from http://www.webopedia.com/TERM/C/CMIP.html

20. Webopedia, 'Middleware'(January 25, 2002) Retrieved April 12, 2002 from http://webopedia.internet.com/TERM/m/Middleware.html

21. Webopedia, 'Transparency'(n.d.) Retrieved May 15, 2002 from http://www.webopedia.com/TERM/n/network_transparency.html