A SURVEY of MIDDLEWARE

Toni A. Bishop
Computer & Information Science Dept
Towson University
Towson, Maryland 21252, USA
tbishop@towson.edu

Ramesh K. Karne
Computer & Information Science Dept
Towson University
Towson, Maryland 21252, USA
rkarne@towson.edu

Abstract

The term middleware has been used to describe many different types of software products. Middleware is essentially supplementary software that connects two or more software together. Since there is wide variety of software included under this big umbrella called middleware, categories need to be assigned to further describe and delineate each of these types. The current literature has provided several ways to classify these packages. These classifications seem to be conflicting and do not cover all the types of middleware available today. Difficulties in classifying middleware are further exasperated by the fact that some middleware can perform more than one service. We will show in this paper, a taxonomy based on the way each middleware either assists an application or helps with the integration of multiple software into a system.

KEY WORDS: Middleware and classification.

1 INTRODUCTION

The word "Middleware" has been used to describe a wide variety of software products. This can cause a problem in understanding exactly what are middleware. In the literature, there are many different middleware definitions. They range from a software layer between the operating system (and/or the network) and application [3] to "glue" between two applications [36]. It has also been described as an important integration tool [7] or enables modular connection to distributed software [2]. A possible definition for middleware is "Middleware is the software that assists an application to interact or communicate with other applications, networks, hardware, and/or operating systems. This software assists programmers by relieving them of complex connections needed in a distributed system. It provides tools for improving quality of service (QoS), security, message passing, directory services, file services, etc. that can be invisible to the user." This definition describes most of the middleware products available today.

Classification is not new in this middleware area. Many authors have suggests classifying by various communication techniques, protocols (session versus sessionless), the programming interface, [5], specific categories included database-oriented, virtual system, middle-tier, gateways, and Web-enabled [25], communication with programming execution and number of participants [32], or ten specific categories [21]. While these classifications cover many aspects of middleware, we will describe a different approach.

2 TAXONOMY OF MIDDLEWARE

In this proposed taxonomy of middleware, there are two major categories: integration type and application type. We further describe each of these subcategories in detail in the following subsections. Figure 1 displays this proposed taxonomy.

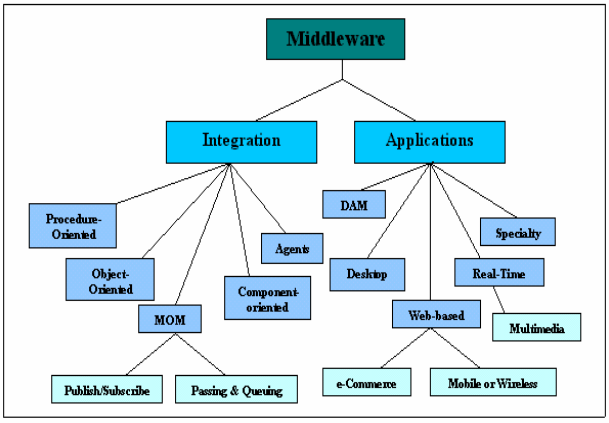

Figure 1: Categories of Middleware

2.1 Integration Category

The integration type includes those middleware that have a specific way of being integrated into its heterogeneous system environment. Each of these middleware has different communication protocols or ways of operating between the other software. We will discuss each of these integration type middleware as follows.

2.1.1 Procedure Oriented Middleware

Procedure-oriented Middleware uses a synchronous communication (like a telephone). The characteristic of the procedure-oriented middleware is the use of client stubs and server skeletons. The client stub converts the parameters of a procedure into a message (known as marshalling) and sending this message to the server (or host). The server skeleton converts the message back into the parameters and sends the procedure call to the server application where it is processed. Once the procedure has been performed the reverse process occurs. The client stub also checks for errors, sends the results to the calling software, and then suspends the thread [3] [13].

The advantages of this middleware are that it uses standard types of name services, remotes processes, can return a response even with network problems [25], supports exceptions [13], and can manage multiple types of data formats and heterogeneous system-level services. The disadvantages include not being scaleable because it contains no replication mechanism, not being reflective (in that it cannot return data to another program) [13] and is a very rigid process in that it is tightly couple to the procedure. [16]

2.1.2 Object Oriented Middleware

The Object-oriented (OO) Middleware supports distributed object requests. The communication between objects (client object and server object) can be synchronous, deferred synchronous, and asynchronous using threading policies. Object middleware supports multiple transactions by grouping similar requests from several client objects into a transaction [13]. The OO middleware operates with the client object first making a logical method call to remote object. A local ORB proxy (also known as a stub) marshals the data and transmits it across to the Broker. The Broker acts as a middleman that may contact a number of data sources, obtains their reference IDs, collecting data, and sometimes reorganizing data [1]. The server object receives the message from the Broker and the remote proxy (also known as the skeleton) unmarshals the data. The data are submitted to a remote servant object where a particular process is performed. The returned of results is performed in reverse from above [23].

The strengths include a support to load management and scalability of the server objects and reflective which were weaknesses to the procedure-oriented middleware. It also operates on many different data types, forms and states and is a self-manager of distributed objects (supports multiple simultaneous operations using multi-threading) [12][24]. The disadvantages include pre-linked prior to execution and the need for wrapper code for some legacy systems [6].

2.1.3 Message Oriented Middleware

This Message Oriented Middleware (MOM) category can be subdivided into two types – Message Passing/Queuing and Message Publish/Subscribe. In the message passing type of this MOM, the application sends out a message (intended for one or more clients) using the client MOM. The MOM server picks the requests off the queue (message broker) in some predetermined order (either FIFO, based on a priority scheme, or based on a load-balancing method). The MOM server acts as a router to the message and does not usually interact with it [19].

The publish/subscribe MOM works slightly differently. This MOM is an event-driven process. If a client wants to participate, it first joins an information bus. Then depending on its function as the publisher, subscriber, or both, it registers an event listener in the bus. The publisher sends a notice of an event to the bus (on the MOM server). The MOM server then sends out an announcement to the registered subscriber(s) that data is available. When the subscriber requests from a specific publisher some data, the request is wrapped in a message and sent to the bus. The bus then sends an event to the publisher requesting the data [33].

2.1.4 Component Based or Reflective Middleware

A component is described as a "program that performs a specific function and is designed in such a way to easily operate with other components and applications" [36]. This middleware is a configuration of components. These components are selected either at build-time or at run-time. The major contribution of this middleware is that it has an extensive component library and component factories, which support construction on multiple platforms [4]. Since the variation (i.e. resource availability, new required protocols, network connectivity) in the distributed computing environment is increasing yearly, software like component or reflective middleware will be needed to provide the ever increasing QoS requirements [22]. The "reflective architectures provide expansion joints where new behaviors can be imposed." With these joints, changes to the operation can be made during runtime [31].

The strengths of this middleware are that it is configurable and re-configurable. Re-configurability can be accomplished at run-time, which offers a lot of flexibility to meet the needs of a large number of applications [4].

2.1.5 Agents

Agents are considered a middleware that consists of several components: entities (objects, threads), media (communication between one agent and another), and laws (rules on agent's communication coordination). Media can be monitors, channels, or more complex types (e.g. pipelines). Laws identify the interactive nature of the agents such as its synchronization or type of naming schemes [8].

An agent is capable of autonomous actions to meet its design objectives. This adaptability of the agent should be generic so that it covers a broad base of strategies, which range from anytime algorithms, load balancing strategies, resource adaptability, and resource unaware applications. [9].

Dr. Wiederhold's OnTo-Agents establishes an agent infrastructure where each part adds information to the whole system. Several tools establish an exchange of information. [37].

The strengths of Agent middleware are that Agents can perform task on behalf of the user and are adaptable to cover a broad range of strategies based on the environment around them. They make decisions as to the best quality for the purpose and no better [9]. However, the complexities, difficulties in understand the operations, and the abundance of manpower required to incorporate them into a system have been the Agents major weakness.

2.2 Application Categories

The application classification includes middleware that fit into specific types of application functions. These middleware (Data Access, Web-based, Real-Time, Desktop, and Specialty) work specifically with an application.

2.2.1 Data Access Middleware

The Data Access Middleware (DAM) is type of middleware is characterized by the interaction of the application with local and/or remote databases (legacy, relational and non-relational), data warehouses, or other data source files. This category of middleware includes transaction processing (TP) monitor, database gateways, and distributed transaction-processing middleware. This middleware can assist with special database requirements such as security (authentication, confidentiality, and access control), protection, and the ACID properties [5]. This database middleware can even perform requested transactions themselves if the DBMS is unavailable or is unable to handle the transactions. Since the data may be retained on more than one database, the middleware (specifically the transaction processing monitor) tracks the progress of each transaction and can request rollbacks when one part of the request fails. The middleware informs the requesting application of the status of the request and passes all returned data. Some middleware even modify the appearance of the returned data in a format that makes the data easier to use by the application or the user [15].

The strengths include communication between multiple sources and databases transparently, conversions of application programming language into the target database(s) acceptable language, and ability to conversion of response set into a format acceptable to the requesting application. This middleware has the ability to query databases directly or communicate with the DBMS [26].

2.2.2 Desktop Middleware

The Desktop Middleware can make variations in the presentation of the data as requested by a user by tracking and assisting applications, manage any transport services (e.g. terminal emulation, files transfer, printing services), and provide backup protection and other operational background functions with minimal disruption [21]. Additional desktop middleware services include graphics management, sorting, character and string manipulation, records and files management, directory services, database information management, thread management, job scheduling, event notification services, software installation management, encryption services, and access control [3].

2.2.3 Web-based Middleware

Web-based middleware assists the user with browsing, uses interfaces that scout ahead to find pages of interest, and discerns user's changes of interest from browsing history [28]. It provides authentication service for a large number of applications [21] and inter-process communication that is independent from underlying OS, network protocols, and hardware platforms [18]. The middleware tightly bound to the net are called application servers because they improve the "performance, availability, scalability, security, information retrieval, and support of collaborative administration and usage." Middleware may connect directly to the application (circumventing HTTP) when this gains better communication between the server and client [15]. Some core services provided by web-based middleware include directory services, e-mail, billing, large-scale supply management, remote data access (to include downloads, program access, and browsing), and remote applications [34].

One of the web's main uses is e-Commerce, which pertains to the communication between two or more businesses (or patrons and businesses) performed over the web. This middleware controls access to customer profile information, allows the operation of business functions such as purchasing and selling items, and assists in the trade of financial information between applications. This business middleware can provide a modular platform to build the next generation of web applications [21]. The need for security, QoS, cost-effective and speedy transactions, and transparency over diverse environments is essential [7].

The strengths of e-Commerce middleware include enabling the fast integration of various computing systems into a web-based business solution, communicating between businesses easier, cost-effective, and more secure, and allowing customer service representatives to access data from multiple customer information systems [7].

Mobile or Wireless middleware is the other main subdivision of this web middleware. It integrates distributed applications and servers without permanently connecting (through wires) to the web. It provides mobile users secure, wireless access to e-mail, calendars, contact information, task lists, etc. [14].

Some key issues of the mobile computing environment include bandwidth, reliability, delay, latency, error rate, user interface, processing power, interoperability, and cost [20]. The bandwidth and error rates increase exponentially in a wireless environment. The mobile system has other problems as well that include battery power fluctuations, network drops, moving out of signal range, location awareness, and the cost of communication. When the signal is transferred from one signal source to another, additional concerns arise like authentication (ensuring that this is the same user) [16] and checking for lost packets.

This wireless connection is accomplished by using a different set of protocols, tools, and procedures than a physically attached computer to the network. To see more on this go to http://www.wapforum.org/ [35].

2.2.4 Real Time Middleware

Real-time is characterized by the right data being provided on time otherwise it is no longer the correct data [29]. The real-time middleware supports time-sensitive request and scheduling policies. It does this with services that improve the effectiveness of the user applications. Real-time middleware can be divided into the different applications using them (real-time database application, sensor processing, and information passing).

Real-time information passing middleware has increased dramatically with the introduction of the Internet, wireless networks, and new "dissemination-based applications".

Strengths of time-dependent middleware are that they provide a decision process that determines the best approach for solving time-sensitive procedures [30] and they can assist the operating system in the allocation of resources to aid time-sensitive applications to meet their deadlines [11].

Multimedia middleware is a major part of the real time category and can reliably handles a variety of data types. These types include speech, images (pictures, GPS outputs, etc.), natural language processors (translators and teleprompters), music, and video. The data need to be collected, integrated, and then delivered to the user in a time sensitive manner [27]. Multimedia systems can integrate a mixture of logical and physical devices. Physical devices may include video editors, cameras, speakers, and processing devices (data encoder/decoders or media synthesizers) [10]. One important time-sensitive multimedia middleware assists in the distributed video-on-demand services. These services require a process to open new connections, ensure payment for services to the provider, and most importantly, ensure the quality of service (QoS) delivered to the customer.

2.2.5 Specialty Middleware

As with most cases of categorization, there are several types of middleware that provide for specific needs that do not fit into the above categories. There were several middleware that fell into this category on the Internet2.edu website. These include Multi-Campus System and Medical middleware [17]. Additional; very specific types of middleware may be assigned to this category [2].

3 CONCLUSION

As stated before the word middleware has been used to encompass many different types of software therefore it is hard to determine exactly what is being referenced. The taxonomy proposed by other authors were discussed and shown to be inadequate in including so many types. A new classification scheme was presented in which middleware are divided into two major grouping (Integration and Applications). These two major classifications were then subdivided into categories based on the mode of operation or the support given to applications. We believe that this taxonomy provides better structure for middleware and provide a basis for future expansion of classifications.

4 REFERENCES

[1] M. Altinel et al, "DBIS-Toolkit: Adaptive Middleware for Large Scale Data Delivery," Proc. *SIGMOD '99*, vol. 28, issue 2, May 1999.

[2] M. Astley, D. Sturman, et al, "Customizable Middleware for Modular Distributed Software," Comm. ACM, vol. 44, issue 5, May 2001.

[3] P. Bernstein, "Middleware", Comm. ACM, vol. 39, issue 2, Feb 1996.

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

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

[6] A. Campbell, G. Coulson, and M. Kounavis, "Managing Complexity: Middleware Explained," IT Pro, Sep/Oct 1999.

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

[8] P. Ciancarini, "Coordination Models and Languages as Software Integrators," ACM Computing Surveys (CSUR), vol. 28, issue 2, Jun 1996.

[9] Y. Ding, et al, "RAJA," Proc. 5th International Conference on Autonomous Agents, May 2001.

[10] D. Duke and I. Herman, "A Standard for Multimedia Middleware," ACM Multimedia'98, Aug 1998.

[11] H. Duran and G. Blair, "A Resource Management Framework for Adaptive Middleware," Proc. 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, Mar 2000.

[12] W. Emmerich, "Distributed Objects," Proc. 1999 Software Engineering, May 1999.

[13] W. Emmerich, "Software Engineering and Middleware: A Roadmap," ACM Proc. on the Future of Software Engineering, May 2000.

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

[15] P. Fraternali, "Tools and Approaches for Developing Data-Intensive Web Applications: A Survey," Computing Survey, vol. 31, issue 3, Sep 1999.

[16] K. Geihs, "Middleware Challenges Ahead," Computer, vol. 34, issue 6, Jun 2001.

[17] Internet2, http://middleware.internet2.edu/.

[18] IT Pro, http://www.itprodownloads.com/.

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

[20] L. Kleinrock, "Breaking Loose," Communications of the ACM, Sep 2001.

[21] KnowledgeStorm, http://www.knowledgestorm.com/, headquartered in Atlanta, GA.

[22] F. Kon, et al, "The Case for Reflective Middleware," Comm. of the ACM, vol. 45, no. 6, Jun 2002.

[23] Y. Krishnamurthy, et al, "Integration of QoS-Enabled Distributed Object Computing Middleware for Developing Next-Generation Distributed Applications," ACM SIGPLAN Notices, vol. 36, issue 8, Aug 2001.

[24] S. Lewandowski, "Framework for Component-Based Client\Server Computing," ACM Computing Survey (SCUR), vol. 30, issue 1, Mar 1998.

[25] D. Linthicum, "Next Generation Middleware," DBMS, Sep 1997, http://www.dbmsmag.com/9709d14.html.

[26] D. Linthicum, "Database-Oriented Middleware," DM Review, Nov 1999, http://www.dmreview.com/editorial/dmreview/print_action.cmf?EdID=1560.

[27] K. Mills, "Introduction to the Electronic Symposium on Computer-Supported Cooperative Work," ACM Computing Surveys (CSUR), vol. 31, issue 2, Jun 1999.

[28] MIT Media Lab: Software Agents: Projects, May 2001, http://agents.media.mit.edu/projects/.

[29] D. Schmidt, "Middleware for Real-Time and Embedded Systems," Comm. of the ACM, vol. 45, no. 6, Jun 2002.

[30] G. Slivinskas, C. Jensen, and R. Snodgrass, "Adaptable Query Optimization and Evaluation in Temporal Middleware," Proc. International Conference on Management of Data on Management of Data, May 2001, vol. 30, issue 2, May 2001.

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

[32] J. Thompson, "Avoiding a Middleware Muddle," IEEE Software, Nov/Dec 1997.

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

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

[35] WAP Forum, http://www.wapforum.org/.

[36] Webopedia Website, http://www.webopedia.com/.

[37] G. Wiederhold, et al., "OntoAgents – a Project in the DARPA DAML Program," http://www-db.stanford.edu/Ontoagents.