ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 1, Issue 2, November 2012

Middleware: An Architecture for Distributed Systems Services

Savita Chavan , Mangal Sonawane, Ashwini Chavan, Madhavi Sarjare
Lecturer, Department of Computer Technology & Information Technology, YTCEM, Mumbai

Abstract- Service-oriented architectures are poised to transform the industrial scene by enabling more flexible and agile IT infrastructures. The key change agent in this transformation is middleware. Mmiddleware optimises the cost and delivery of IT services.Middleware's primary roles are first functionally bridge the gap between application programs and the lower-level hardware and software infrastructure to coordinate how parts of applications are connected and how they interoperate and second is provide reusable services that can be composed, configured, and deployed to create distributed systems rapidly and robustly by integrating components that may be developed by multiple technology suppliers.

***Index Terms*— Middleware Distributed Operating System, RPC, Database, Objects.**

I. INTRODUCTION

Middleware as the name suggests, sits in between the Operating System and the Application programs. The term middleware applies to a software layer that provides

- A programming abstraction
- Masks the heterogeneity of the underlying networks, hardware, operating systems and programming languages.

Middleware is software glue. Middleware is the slash in Client/Server.

Middleware is an important class of technology that is helping to decrease the cycle-time, level of effort, and complexity associated with developing high-quality, flexible, and interoperable distributed systems. Increasingly, these types of systems are developed using reusable software (middleware) component services, rather than being implemented entirely from scratch for each use. When implemented properly, middleware can help to:

Shield developers of distributed systems from low-level, tedious, and error-prone platform details, such as socket-level network programming.

• Amortize software lifecycle costs by leveraging previous development expertise and capturing implementations of key patterns in reusable frameworks, rather than rebuilding them manually for each use.

• Provide a consistent set of higher-level network-oriented abstractions that are much closer to application and system requirements to simplify the development of distributed systems.

• Provide a wide array of developer-oriented services, such as logging and security that have proven necessary to operate effectively in a networked environment.

Middleware was invented in an attempt to help simplify the software development of distributed computing systems, and bring those capabilities within the reach of many more developers than the few experts at the time who could master the complexities of these environments. Complex system integration requirements were not being met from the application perspective, where it was too hard and not reusable, or the network or host operating system perspectives, which were necessarily concerned with providing the communication and end system resource management layers, respectively.

Over the past decade, middleware has emerged as a set of software service layers that help to solve the problems specifically associated with heterogeneity and interoperability. It has also contributed considerably to better environments for building distributed systems and managing their decentralized resources securely and dependably. Consequently, one of the major trends driving industry involves moving toward a multi-layered architecture (applications, middleware, network and operating system infrastructure) that is oriented around application composition from reusable components, and away from the more traditional architecture, where applications were developed directly atop the network and operating system abstractions. This middleware-centric, multi-layered architecture descends directly from the adoption of a network-centric viewpoint brought about by the emergence of the Internet and the componentization and commoditization of hardware and software.

Infrastructure that supports middleware are (distributed) component based application development

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 1, Issue 2, November 2012

- Distributed component platforms
- Mechanisms to enable component communication
- Mechanisms to hide distribution information
- Set of predefined components

The middleware Standard for constructing and interconnecting components are Interchange, upgrade, adaptation, and aggregation.

II. BACKGROUND

How did middleware evolve from necessary evil— proprietary plumbing that glued disparate systems together— to one of the most strategic areas of IT and business today?

Because businesses, institutions, and technologies change continually, the software systems that serve them must be able to accommodate such changes.

Following a merger, the addition of a service, or the expansion of available services, a business can ill afford to recreate its information systems. It is at this most critical point that it needs to integrate new components or to scale existing ones as efficiently as possible. This layer, called middleware, allows software components (applications, enterprise java beans, servlets, and other components) that have been developed independently and that run on different networked platforms to interact with one another. It is when this interaction is possible that the network can become the computer.

Service-oriented architectures get many of their core services directly from middleware, including critical security functionality, deployment and management capabilities. One also finds business intelligence, content and collaboration tools, as well as portal capabilities that allow connections to customers and partners enabled at the middleware level. The convergence of these critical business services at the middleware layer is reflective of the mid-tier's strategic position within the enterprise. The broad range of capabilities offered by today's middleware products enables industry to:

- Support and accelerate business expansion
- Deliver greater insight into business issues and drivers
- Reduce exposure to risk and support governance initiatives.

III. NEED FOR MIDDLEWARE

In the age of computers, each user seeking knowledge has a desktop appliance that connects to the utility, i.e. client asking server for the information.

Here desktop appliance can be computer or devise like computer e.g. a terminal, personal computer, workstation, word processor etc. The utility is an enterprise wide network of information services which includes applications, databases on LAN and WAN. Servers on LAN support files and file based applications, such as e-mail, bulletin board, document preparation and printing. They also support directory services. Directory services help desktop users to search other users and services of their interest. Servers on WAN generally support database access. Database access includes corporate directories and electronic libraries or transaction processing applications purchasing, billing and inventory control.

Most of the organizations have large variety of heterogeneous hardware systems which include personal computers, workstations, minicomputers and main frames. All these heterogeneous systems use different OS and network architecture. So integration of these systems is difficult. Here comes the middleware in picture. Middleware deals with providing environments for developing systems that can be distributed effectively over a variety of topologies, computing devises and communication network. It aims to provide developers of networked applications with the required platform and tools to

- Formalize and coordinate how parts of applications are composed and how they interoperate.
- Monitor, enable and validate the configuration of resources to ensure appropriate application service quality, in case of failure also.
- Network communication

- marshaling/unmarshalling

- Coordination

- Activation/termination, threading, group requests, synchronicity

- Reliability

- Delivery guarantees, total/partial ordering, atomicity, replication

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 1, Issue 2, November 2012

- Scalability

- Transparency of access/location/migration/ replication, load balancing

- Heterogeneity

- Platform, operating system, network OS, programming language.

A. ELEMENTS OF MIDDLEWARE

- Software components

- Component interfaces such as properties, methods and events.

- Containers

- shared context of interaction with other components

- provide access to system-level services

- Metadata

- self-descriptive information used by a component to flexibly communicate with others

- Integrated development environment

- e.g., VisualCafe for Java

IV. MIDDLEWARE ARCHITECTURE

Middleware has a layered structure like a networking protocol.

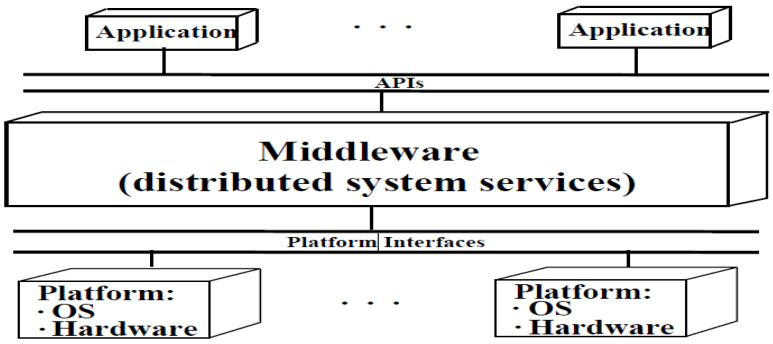

Fig 1: Middleware

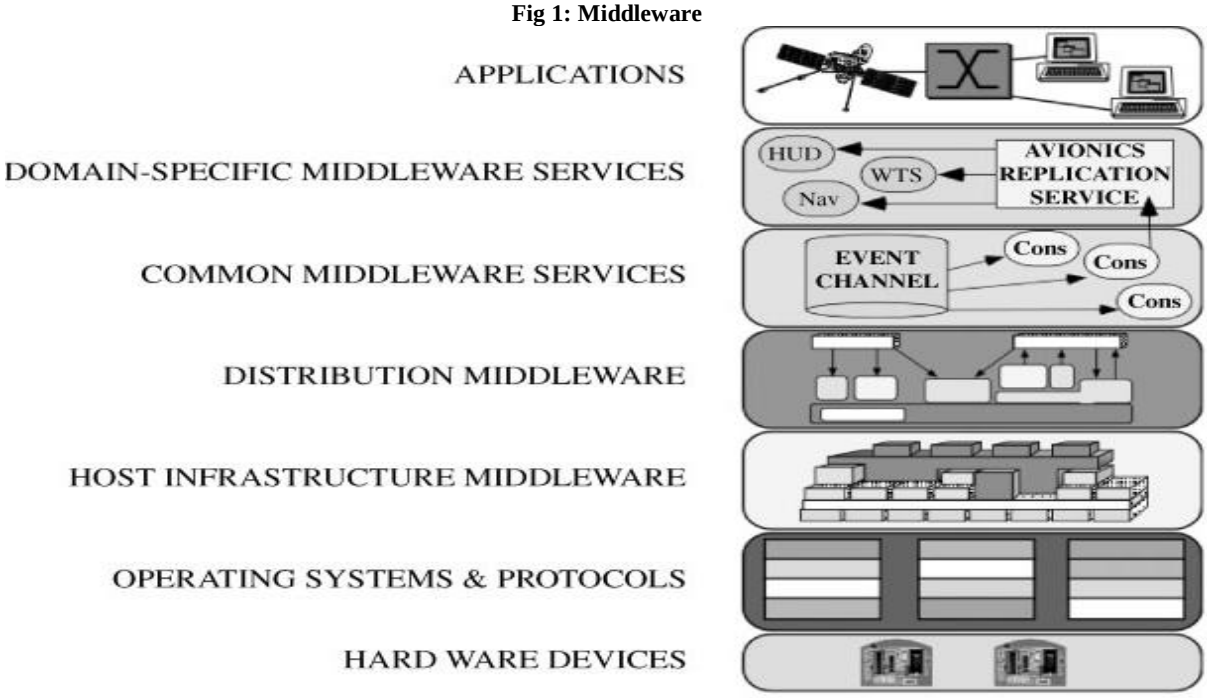

Fig 2: Layers of Middleware and Surrounding Context

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 1, Issue 2, November 2012

V. TYPES OF MIDDLEWARE

A. Message Oriented Middleware

This is a large category and includes communication via message exchange. It represents asynchronous interactions between systems. It reduces complexity of developing applications that span multiple operating systems and network protocols by insulating the application developer from details of various operating system and network interfaces.

APIs that extend across diverse platforms and networks are typically provided by the MOM. MOM is software that resides in both portions of client/server architecture and typically supports asynchronous calls between the client and server applications. Message queues provide temporary storage when the destination program is busy or not connected. MOM reduces the involvement of application developers with the complexity of the master-slave nature of the client/server mechanism. E.g. Sun's JMS.

B. Object Request Broker

In distributed computing, an object request broker (ORB) is a piece of middleware software that allows programmers to make program calls from one computer to another via a network.

ORB's handle the transformation of in-process data structures to and from the byte sequence, which is transmitted over the network. This is called marshaling or serialization. Some ORB's, such as CORBA-compliant systems, use an Interface Description Language (IDL) to describe the data which is to be transmitted on remote calls. E.g. CORBA

C. RPC Middleware.

This type of middleware provides for calling procedures on remote systems, so called as Remote Procedure Call. Unlike message oriented middleware, RPC middleware represents synchronous interactions between systems and is commonly used within an application.

Thus, the programmer would write essentially the same code whether the subroutine is local to the executing program, or remote. When the software in question is written using object-oriented principles, RPC may be referred to as remote invocation or remote method invocation. Client makes calls to procedures running on remote systems, which can be asynchronous or synchronous. E.g. DCE RPC.

D. Database Middleware.

Database middleware allows direct access to data structures and provides interaction directly with databases. There are database gateways and a variety of connectivity options. Extract, Transform, and Load (ETL) packages are included in this category. E.g. CRAVE is a web-accessible JAVA application that accesses an underlying MySQL database of ontologies via a JAVA persistent middleware layer Chameleon).

E. Transaction Middleware.

This category as used in the Middleware Resource Center includes traditional transaction processing monitors (TPM) and web application servers. e.g. IBM's CICS.

F. Portals

Enterprise portal servers are included as middleware largely because they facilitate "front end" integration. They allow interaction between the user's desktop and back end systems and services. e.g Web Logic.

Table 1.COMPARISON OF VARIOUS TYPES OF MIDDLEWARE

Middleware Type	Communication	Processing	Storage
Database Middleware	Yes	Limited	Yes
Remote Procedure Call	Yes	Yes	No
Message-Oriented Middleware	Yes	No	Limited
Distributed Objects	Yes	Yes	Yes
Portals	Yes	Yes	Yes

VI. EXAMPLES OF MIDDLEWARE

The term middleware is used in other contexts as well. Middleware is sometimes used in a similar sense to a software driver, an abstraction layer that hides detail about hardware devices or other software from an application.

1. The Android environment uses the Linux operating system at its core, and also provides an application framework that developers incorporate into their applications. In addition, Android provides a middleware layer including libraries that provide services such as data storage, screen display, multimedia, and web browsing. The Android middleware layer also contains the Dalvik virtual machine and its core Java application libraries.
2. Game engine software such as Gamebryo and Render ware are sometimes described as middleware, because they provide many services to simplify game development.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 1, Issue 2, November 2012

3. In simulation technology, middleware is generally used in the context of the high level architecture (HLA) that applies to many distributed simulations. It is a layer of software that lies between the application code and the run-time infrastructure.
4. Wireless networking developers can use middleware to meet the challenges associated with wireless sensor network (WSN), or WSN technologies. Implementing a middleware application allows WSN developers to integrate operating systems and hardware with the wide variety of various applications that are currently available.
5. The QNX operating system offers middleware for providing multimedia services for use in automobiles, aircraft and other environments.
6. Multimedia Home Platform (DVB-MHP) is an open middleware system standard designed by the DVB project for interactive digital television. The MHP enables the reception and execution of interactive, Java-based applications on a television set.

VII. PROGRAMMING WITH MIDDLEWARE

Programmers do not have to learn a new programming language to program middleware. There are three main ways in which middleware can be programmed with existing languages.

- The first is where the middleware system provides a library of functions to be called to utilize the middleware; distributed database systems
- The second is through an external interface definition language (IDL). In this approach, the IDL file describes the interface to the remote component, and a mapping from the IDL to the programming language is used for the programmer to code to.
- The third way is for the language and runtime system to support distribution. Natively; for example, Java's Remote Method Invocation (RMI).

VIII. CONCLUSION

Middleware has become inseparable part of the almost all applications now days. This is mainly due to widespread use of Information Technology and the emerging trend of distributed computing.
Middleware's are user friendly, requires no additional language learning.

REFERENCES

[1] Bernstein P., "Middleware An Architecture for Distributed System Services," CACM, 39:2, March 2, 1993.

[2] Blair, G.S., F. Costa, G. Coulson, H. Duran, et al, "The Design of a Resource-Aware Reflective Middleware Architecture", Proceedings of the 2nd International Conference on Meta-Level Architectures and Reflection, St.-Malo, France, Springer-Verlag, LNCS, Vol. 1616, 1999.

[3] Bernstein, P., "Middleware, A Model for Distributed System Service"", Communications of the ACM, 39:2, February1996.

[4] Blair G.S., Costa F., Coulson G., Duran H., etal, "The Design of a Resource-Aware Reflective Middleware Architecture", Proceedings of the 2nd International Conference on Meta-Level Architectures and Reflection, St.-Malo, France, Springer-Verlag, LNCS, Vol. 1616, 1999.

[5] Kaoutar El Maghraoui, Travis J. Desell, Boleslaw K. Szymanski, and Carlos A. Varela," The Internet Operating System: Middleware for Adaptive Distributed Computing".

[6] Cukier M., Ren J., Sabnis C., Henke D., Pistole J., Sanders W., Bakken B., Berman M., Karr D. Schantz R., "AQuA: An Adaptive Architecture that Provides Dependable Distributed Objects ", Proceedings of the 17th IEEE Symposium on Re-liable Distributed Systems, pages 245-253, October 1998.

[7] Foster, I. and Kesselman, K., "The Grid: Blueprint for a Future Computing Infrastructure," Morgan Kaufmann, 1999.

[8] Beck K., eXtreme Programming Explained: Embrace Change, Addison-Wesley, Reading, MA, 2000.

[9] Bollella, G., Gosling, J. "The Real-Time Specification for Java," Computer, June 2000.