MEDINFO 98
B. Cesnik et al. (Eds)
Amsterdam: IOS Press
© 1998 IMIA. All rights reserved

Middleware for Healthcare Information Systems

Stéphane Spahni[a], Jean-Raoul Scherrer[a], Dominique Sauquet[b], Pier-Angelo Sottile[c]

[a]*Division d'Informatique Médicale, Hôpital Cantonal Universitaire de Genève Genève, Suisse.*
[b]*Broussais University Hospital, Paris, France*
[c]*GESI / Gestione Sistemi per l'Informatica SRL, Roma, Italy*

Abstract

Middleware is now a commonly used expression and anyone building distributed applications is referring to "middleware services". Nevertheless this notion lacks of sound theoretical foundation. This paper tries to clarify the relationship between the components of distributed environments, especially in healthcare, and to establish some classification aiming at gaining a common understanding of the functionalities and interdependency of the existing modules of distributed environments.

Keywords
Middleware; Healthcare Information Systems ; OSI

Introduction

As stated by P. A. Bernstein in ACM communication [1], the computing facilities of large enterprises are evolving into a utility. This is especially true for Healthcare Information Systems (HIS), which should become far more portable for one site to another in order to ‥.ıit development and maintenance costs.

Each application terminal (e.g. a PC, a workstation) is a desktop appliance that connects to the utility, the utility itself being an enterprise-wide network of information services including applications and databases. Servers on the local area network support applications such as electronic mail, board minutes, document preparation, and printing as well as access to DBS (Database System) or transaction processing applications. Some servers are gateways to services and could offer electronic document interchange between institutions or equivalent.

To help solve users heterogeneity and distribution problems, information utilities that have standard programming interfaces and services are called middleware (MW) services, because they sit in above the operating system and networking software and below specific applications. For many new applications, middleware components are becoming more important than the underlying OS and networking services on which the application services formerly depend. As an example, new applications often depend more on a SQL DB (Database Manager) rather than on an OS's record oriented file system and more on an RPC (Remote Procedure Call) mechanism rather than on transport - label messaging.

A "middleware service" is a general purpose service that sits between "platforms" and applications. A "platform" here is a set of low level services and processing elements - defined by a processor architecture and an operating system's API (Application Programme Interface). It therefore implements functionality of general interest for complex distributed systems, and in particular for healthcare distributed systems.

However the present situation is messy! There are as many different ways of doing things as there are implementations of distributed information systems using varied pathways, e.g. as indicated in Figure 1. It is thus highly desirable to consensually restrict the commendable pathways to only the most appropriate ones deduced from the critical analysis of the present situation.

The concept of the "Middleware" approach

The design concept

Middleware is quite a fuzzy notion, often used without any reference to a sound theoretical foundation. However, when building structured applications, distributed ones in particular, it is difficult not to deal with this middleware approach. Although the literature about middleware is relatively scarce [1-2], several common characteristics may be identified straight away:

- The middleware is at the application service level: then it implements services for high level applications. The common functionalities are grouped into a "layer" called middleware, enabling the developers to concentrate on the application's specific concerns.
- The middleware is strongly connected to distributed systems: then it addresses essentially complex applications spread over many systems, i.e. used in a distributed environment.
- The middleware runs on different platforms: then the middleware evidently has to be available on heterogeneous platforms in order to support independence between the environment and the applications. The most commonly used environment for distributed applications is based on client/server approach, where the servers are typically Unix or Windows NT™ servers and clients PCs, Macintoshes or even Unix workstations in the 3-tier architecture [3].

- The middleware takes advantage of bitways or DBMS services: it relies on lower level bitways services for gaining access to the network, and possibly uses DBMS services for getting access to database systems.

The middleware aims therefore at reducing the impact of problems related to the development of complex applications within heterogeneous environments, offering high level standardized services hiding most of this heterogeneity. Indeed the concept itself of middleware has its origin in the development of distributed systems, where the applications run on a sort of "cloud of systems" and are therefore clients of good and easy-to-use information exchange facilities implementing the required communication between processes.

How standardized functionalities are implemented into the "middleware"

If, for instance, portability, interoperability and wide availability terms are related to standards, then it appears that middleware is quite often subject to standardization.

In general, two categories of standards are identified: de facto standards and expressed standards. De facto standards are mainly issued by software companies while expressed ones are issued by standardization bodies like the International organization for standardization (ISO), the American National Standard Industry (ANSI), the International Union of Telecommunications (ITU), and others. The main difference between the two categories comes from the way they are promoted. Expressed standards are defined by groups of people, and when a consensus is met, the standard is published and implementations are then expected to be achieved! De facto standards are based on already implemented products, widely accepted and with already a certain level of robustness and of performance. When the product becomes widely enough available and used, it becomes a standard!

Middleware standards can be found in both categories, although most of them are de facto standards as it will be noticed in the next chapter. But there is quite a significant amount of work being done for example in the medical field for establishing expressed standards.

Principal trends in the development of MW

Middleware is being developed since several years: some common characteristics can therefore already be identified for establishing a classification of the various trends in the developments. One of the most significant is the level of abstraction of the services provided by the middleware. Three classes or generations can already be defined:

First generation: Elementary Middleware

The first generation of middleware dates from the mid-eighties, when distributed systems really became "the" way for replacing ageing mainframes. Basic tools, called "Remote Procedure Call" (RPC), were developed for enabling applications to delegate part of their work to other systems. This quite rudimentary way of delegating tasks implies that the remote machine knows some pre-defined functions and is able to process requests to these functions with parameter values received from the network, the results being sent back the same way. It has to be noted that although the concept of RPC is widely accepted, there are different and generally incompatible implementations (e.g. Sun RPC, DCE RPC (Distributed Computing Environment), OSI (Open Systems Interconnection) RPC, etc.).

Extensions to the RPCs rapidly became available in order to govern transparent access to distributed databases. "Transparency" refers here both to the location of the database when networked databases are being accessed and to the real database manager being used (e.g. Ingres, Oracle, DB2, etc.), "folklore" into which we are reluctant to enter since it is against the concept of openness! The SQL language is an example of a middleware-level interface enabling this independence against product-specific query languages.

Second generation: Middleware as a support of distribution

With the development of more and more complex and varied distributed systems came a growing demand for extended services handling the distribution (dynamic or not) of tasks, for localizing resources over the network, or for managing share common resources. This resulted in the development of *middleware environments*, from which one may mention DCE, PVM (Parallel Virtual Machine), or TUXEDO®.

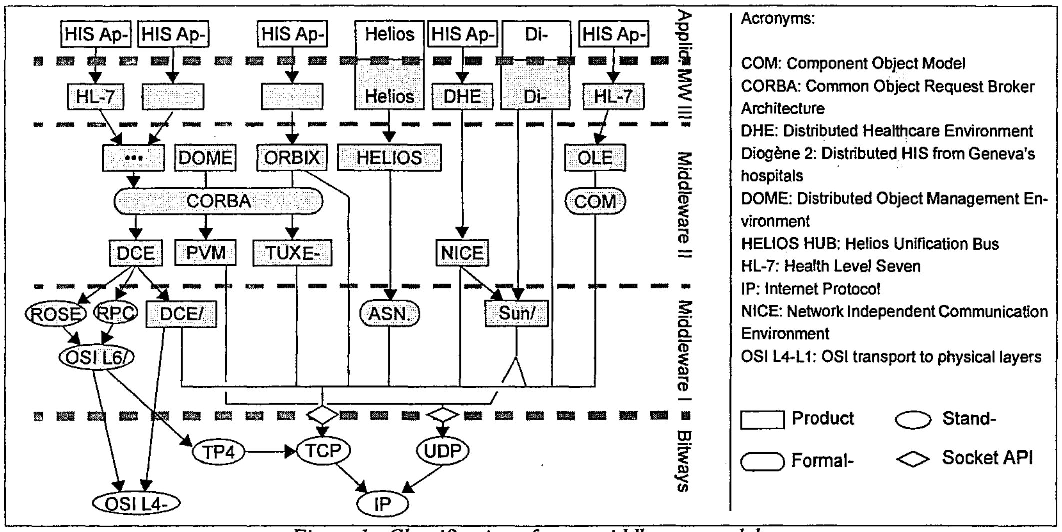
Although the complexity and the services of these environments differ with each other, they all offer various more advanced services compared to those offered by the middleware of the first generation. DCE, for example, offers in addition to basic RPC time synchronization between systems, resources sharing and a directory service.

These tools have now been enhanced in order to take into account the growing use of object-oriented concepts. This led to expressed standards on object technology (e.g. CORBA, the Common Object Request Broker Architecture) [4] and to object oriented environments like HELIOS [5], Microsoft® OLE (Object Linking and Embedding) or ORBIX-TUXEDO® [6] to mention only some of the main ones.

The second generation of middleware, as well as the first, enhances the portability of applications over different platforms. However, the reusability of an application in another environment is still not easy to achieve as many specific services depend on the type of organizational environment in use. Healthcare information systems are a good example of one type of such organisational environments. This concern led to the latest generation of middleware, tailored to specific domains of activity.

Third generation: Domain-specific Middleware

The middleware has to provide generic services to applications, which does not mean that the applications have to be generic! We can therefore classify under the name "third generation of middleware" what is being developed now: middlewares tailored to specific environments. The goal of these environments is to offer generic services in one specific domain of activity, grouping the domain-specific tasks and information common to all applications into the middleware layer. Through the identification and isolation of high level but domain-specific tasks, a higher level of portability can be reached, and applications can



Acronyms:

COM:	Component Object Model
CORBA:	Common Object Request Broker Architecture
DHE:	Distributed Healthcare Environment
Diogène 2:	Distributed HIS from Geneva's hospitals
DOME:	Distributed Object Management Environment
HELIOS HUB:	Helios Unification Bus
HL-7:	Health Level Seven
IP:	Internet Protocol
NICE:	Network Independent Communication Environment
OSI L4-L1:	OSI transport to physical layers

Figure 1 - Classification of some middleware modules

be easily ported from one organization to another. It can be noted that the difference between the second and the third generation can be easily compared to the difference between objects and specialized objects, or "objets métier".

Developments currently made in the field of healthcare are good instantiations of such middleware services. Several research projects are exploring desired features of Health Information System's Middleware (HIS middleware) in order to facilitate the exchange of data between hospitals and to make possible the reuse of the same applications in several sites. As example, one can mention the work of the Health Level Seven group (HL-7) - accredited by the ANSI to write the standard of the same name [7], or the SYNAPSES project briefly described afterwards.

As a summary, the Figure 1 presents a refinement of the 3 layers model (bitways, middleware, application) commonly used to describe the tools and protocols for building an application: the middleware layer is divided into 3 sub-layers, reflecting the classification presented above. One can see that most of the (n+1) generation middleware is based on services from the (n) generation. It has to be noted that Figure 1 is not intended with the idea of being exhaustive regarding the products listed, but rather wants to present graphically the various abstraction levels making up the protocol stack of a HIS (Healthcare Information System / Hospital Information System) application.

Middleware and the OSI model

As seen in the previous chapter, there are many middleware environments, with different levels of functionality. Some formalization of the structure and the relationships between these environments becomes mandatory, in order to make possible structured descriptions of the applications and the services being used. To help establishing this structure we refer to the Open Systems Interconnection Reference Model (OSI RM) defined by the International Organization for Standardization (ISO). This model defines 7 hierarchically structured layers, numbered 1 to 7. The layers can be divided in two groups: the lower layers group, responsible for the transmission, is formed by layers 1 to 3; the upper layers group, session to application layers, contains layers 5 to 7. In between is the transport layer, or layer 4, providing the necessary glue between the network-oriented lower layers and the application-oriented upper layers.

It is important to note that despite its name, the "application" layer is concerned by *services* for applications, and does not contain any application! These can be considered as forming the 8th layer, not addressed by the OSI RM.

The figure establishes a first intuitive mapping between the 3 layers model defined by the middleware approach and the 7 layers OSI model. The equivalence between most of the layers is evident: the applications based on the middleware are above the OSI upper layers, and the end-to-end transmission of bits and bytes is performed by the OSI lower layers.

The parallel between the structure of the middleware and the upper layers can be further expressed by taking into account the specific structure of the OSI application layer and its notion of Application Service Objects. It is then possible to describe the relationships between the middleware components, allowing the necessary understanding of their behaviour. While this can be considered as a necessary step towards the establishment of a library of HIS-related middleware services, it can also be used for determining where and how parallelism could enhance the performances of the middleware stack. For a more detailed description, see [9]

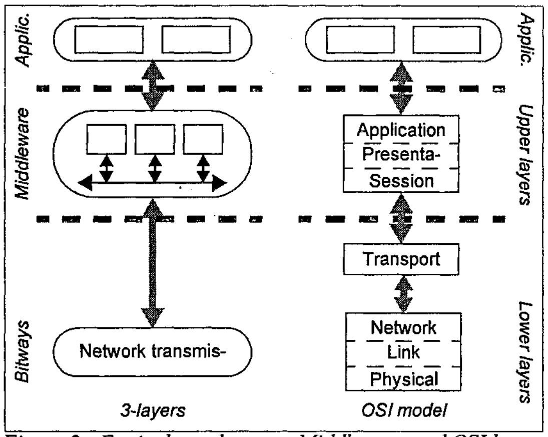


Figure 2 - Equivalence between Middleware and OSI layers

A case study: The SYNAPSES Project

Hospital Information Systems are particularly concerned by the problem of reuse of software. This is emphasized by the necessary migration towards complex distributed systems, where many applications are sharing and exchanging data. Several european projects are therefore developing strategies and standards in order to develop common distributed architectures and make the reuse of software components possible. In particular, the European project SYNAPSES aims at promoting the middleware approach for HIS applications and at establishing standard interfaces for accessing HIS middleware services - typically third generation middleware. According to the standardization terminology, this amounts to developing formulated standards and immediately realizing prototypes assessing the fitness of the standards.

The SYNAPSES project will not reinvent the wheel: it rather tries to capitalize on existing or ongoing work wherever it is possible. Figure 3 is an illustration of this strategy, where the components of a HIS system are represented according to the structure standardized by the Project Team 1-013 of the Comité Européen de Normalisation (CEN) Technical Committee 251 [10]. This standard defines three layers of abstraction: applications, common components and bitways, respectively represented in Figure 3 by applications and services, middleware and bitways.

With respect to Figure 3, one goal of the SYNAPSES project is to define the internal architecture, the functionality and the interfaces of the "Middleware" elements specific to the Healthcare record architecture ("HCC" boxes in the above figure). Fully standardizing such middleware services implies the definition of two interfaces: the interface between the HIS client and the HIS Middleware Services (i.e. the "MW Client Interface") makes possible the development of portable *applications*, while the interface between HIS Services and the HIS Middleware Services is concerned with portability of the middleware itself (and possibly of the HIS services, but in the usual environment where such standards are deployed, changing the HIS services is of no concern)

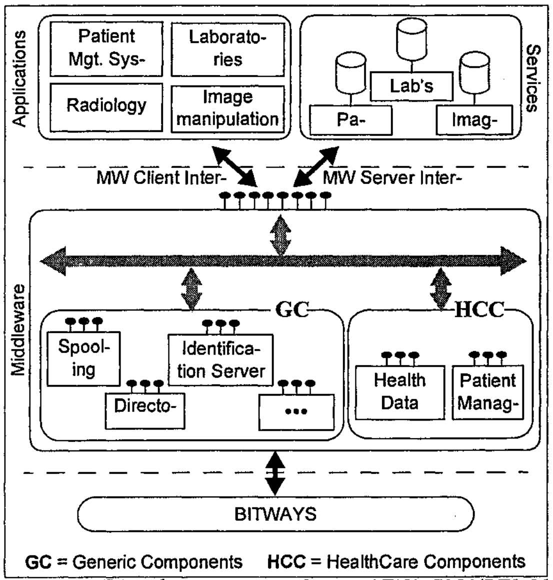


Figure 3 - HIS architecture according to CEN/TC251/PT1-013

The main result of the project will be therefore a high level of portability of applications and of the middleware itself, reached through the standardization of these two interfaces and the definition of a common Federated Health Care Record - expressing the available data using object-oriented technology.

Why use Middleware in Healthcare Information Systems ?

Middleware-based approach for developing software environments in any category of information systems including HIS can be seen just as an nice way of decomposing and classifying functions. But it also offers strong advantages:

- The most important element of a computing environment is the applications. Thanks to standard middleware, the focus of the developments can be primarily user-oriented, what is especially valuable in healthcare centres where the users are changing regularly.
- Sharing of applications is becoming crucial for most organizations, as they cannot afford any more to develop all applications by themselves. By the clear definition of APIs to middleware services, a higher portability of both applications and servers can be reached, enabling thus their reuse in other organizations leading eventually to the establishment of a library of middleware modules.
- In environments like SYNAPSES, the sharing of data is a crucial activity in order to reach a certain level of cost-effectiveness in healthcare. At the University Hospital of Geneva, where more than 50 heterogeneous databases coexist, developing specific tools for enabling each application to access many of these different database managers is unfeasible. The only realistic solution is to use a SYNAPSES-like approach, hiding the heterogeneity into the application services layer, i.e. the middleware layer.

The increasing consensus on the importance of the middleware layers for the migration of large Healthcare Information Systems - with a special focus now on specialized middleware - proves that it is more than just the actual way of thinking. But while the pathways across the first and second generations of middleware are now clearly identified and standardized, it is not yet the case with the third one. There is therefore a risk of getting n^2 solutions, asking at the end for middleware for interconnecting middleware! In order to avoid this, it is vital to promote agreed or de facto standards so that they are adopted by the widest possible audience. This is the actual challenge of the specialized middleware.

Conclusion

Most large organisations view the evolution of their information technology infrastructure as evolutionary, not revolutionary. This translates into the requirement to support heterogeneous configurations, in which applications on proprietary legacy systems interact with new ones.

Distributed systems support the actual business trends and provide direct answers to most of the needs of the market and the users. They provide direct support for decentralised business units, making use of their own local processing, while being given access to company-wide information systems. They also promise to maximize the use of networked resources, services and databases, leading to a minimisation of costs.

New applications should be seen independently of the physical structure of the system so that users finally perceive open systems as an opportunity to obtain equivalent products from several suppliers. Federation is seen to be the key structuring principle to combine components of a system. These issues are nonspecific to medicine and healthcare!

Besides, regarding the client-server distributed new information systems, there are more and more standardized application services. These new services, like for instance the patient identification server, the archive server, etc. constitute a kind of superior middleware which on the one hand make domain-specific applications more easily portable from one environment to another and on the other allow the interoperability with other heterogeneous sub-systems.

Hiding heterogeneity by using specialized middleware is now a widely agreed trend, but there is a serious lack regarding methods and tools for describing the various middleware components and their relationships. The importance of the present work is therefore also in the formalisation in a generic manner of middleware elements along with their mutual relationships. Thanks to the approach adopted here, the progressive construction of HIS from the same common elements is made easier and the comparison of systems becomes possible, which will be highly valuable for avoiding the n^2 approach in interconnecting heterogeneous distributed systems.

Acknowledgments

This study was supported by the SYNAPSES project funded by the European Union (Telematics in Healthcare program, contract no. HC 1046). The Swiss part of the project is fully supported by the Swiss government (Federal office for Education and Science: contract no. OFES 95.0286).

References

[1] Bernstein PA. Middleware: A Model for Distributed System Services. *Com. of the ACM* Feb. 1996: vol. 39/2, pp. 86-98.

[2] King SS. Middleware! *Data Communications* Mar. 1992, pp. 58-67.

[3] Scherrer JR, Baud R, de Roulet D. Moving towards the future design of HIS: A view from the seventies to the end of the nineties, the DIOGENE paradigm. In H.U. Prokosch, J. Dudeck (eds), Hospital information systems: design and development characteristics; impact and future architecture. Elsevier, 1995:347-375.

[4] *PTC/96-03-04: CORBA 2.0 Specification.* Object Management Group Inc., Jul. 1995.

[5] The HELIOS Software Engineering Environment. *Computer methods and programs in biomedicine suppl.* Elsevier, Dec. 1994; 45:S1-S152.

[6] http://www.iona.com/Orbix/index.html

[7] Rishel W. Pragmatic Considerations in the Design of the HL7 Protocol. *Proc 13th AMIA Annu Fall Symp*; Washington DC Nov. 5-8 1989, pp. 687-691.

[8] Spahni S. *Parallélisation des couches supérieures du modèle OSI: stratégies et mise en oeuvre.* Thèse de doctorat no 2588, Genève: Editions systèmes et information, 1993.

[9] Spahni S, Scherrer JR, Sauquet D, Sottile PA. Consensual trends for optimizing the constitution of middleware. *Submitted to ACM Computer Comm. Review.*

[10] *ENV 12967-1:1997 Healthcare Information System Architecture Part 1 (HISA) Healthcare Middleware Layer.* CEN TC/251 PT 1-013, March 1997.

[11] Appel RD, Bairoch A, Hochstrasser DF. Trends in Biochemical Sciences (TIBS), 1994, 19:258-60

Address for correspondence

Dr. Stéphane Spahni, Division d'informatique médicale, Hôpital cantonal universitaire de Genève, rue Micheli-du-Crest 24, 1211 Genève 4, Switzerland.

E-mail: stephane.spahni@dim.hcuge.ch