COMMUNICATING APPLICATION MANAGEMENT : A SERVICE-ORIENTED SERVER MODELLING

Fabienne FAURE[*] [**], Thierry DESPRATS[*], Yves RAYNAUD[*]

[*] I.R.I.T./S.I.E.R.A. Université Paul Sabatier
118, Route de Narbonne, 31062 TOULOUSE Cedex, FRANCE
Tel : (33) 61 55 67 68
Fax : (33) 61 52 14 58
E-mail : faure@irit.fr

[**] HEWLETT-PACKARD FRANCE
Boulevard Steve Biko, 38090 Villefontaine, FRANCE
Tel : (33) 74 82 30 00

Abstract

This paper expresses the foundations of a property which aims at making entirely compatible various computers and softwares, and which is a condition to the achievement of any open distributed processing in the nextcoming years: **interoperability**. We define interoperability from three complementary levels: *communication support, transparency of exchanged data and accessibility to available services*. In relation to these three factors, we establish an overwiew of existing concepts and solutions that nowadays comply with interoperability requirements. The second part of the paper proposes a solution for monitoring of services used by end-users in a communication environment : a **service-oriented management system**. The described system aims at both providing end-users with information about accessible and available services and ensuring quality of provided services. We expose in this paper the basic architectural principles of our solution, and a general specification of the components.

Key words

Interoperability - Service - Service monitoring - Quality of communication system
Service-oriented management system

1. INTRODUCTION

In the context of heterogeneous network management and distributed processing, we deal with the problem of service management. More precisely, we consider end-user (application or user) requirements from a service viewpoint. These services may be typical OSI, Internet or DCE (OSF) services, or any other activity that satisfies a communication need. We are interested in management of configurations of used services, called profiles. Provide with end-users means to query accessible services and means to obtain the best initial profile of services, and subsequently ensure the dynamic monitoring of this profile, are the questions which guide our researches. This service-oriented approach aims at contributing to quality of provided services, for interoperability of communicating entities.

This paper expresses the foundations of a property which aims at making entirely compatible various computers and softwares, and which is a condition to the achievement of any open distributed processing in the nextcoming years: **interoperability**. Then, we focus on the problems of service accessibility and service monitoring which are required in a communication. We propose a ***service-oriented management system*** which solves these probems and thus contibutes to configuration and performance management functionalities relevant to open system management.

2. INTEROPERABILITY, CONDITION FOR OPEN SYSTEMS

By definition, an open system is one in which the components and their composition are specified in a non-proprietary environment, enabling this system to interoperate with any other peer system. The opening degree of a system can be viewed from three perspectives: **portability, integration and interoperability [1]**.

Portability refers to the ability of a system component to be used in various environments (i.e various vendors).

Integration concerns the consistency of the various human-system interfaces or APIs (Application Programming Interfaces). It allows a system component to be used in an uniform way whatever its geographical and hardware locations may be.

Interoperability is the ability of two peer entities involved in a communication to exchange information in a sensible way and to use, each of them with success, the functionalities of the other one [2]. Interoperability is based on the mutual understanding of cooperating entities. It aims at deleting the entity heterogeneity in order to supply with entities a common platform of comprehension and interaction.

Three necessary and complementary interoperability factors (Fig 1) are observed:

* the ***communication support*** (related to the transfer of informations),

* the ***transparency of exchanged data*** (related to representation and comprehension of exchanged data),
* the ***accessibility to available competencies*** of the communication environment (related to the necessary awareness of services for performing a communication).

This classification makes more accurate the previous definition of interoperability: *two end users (application processes or users) are interoperable if they exchange information in a sensible way and if each of them uses with success the functionalities of the other one.*

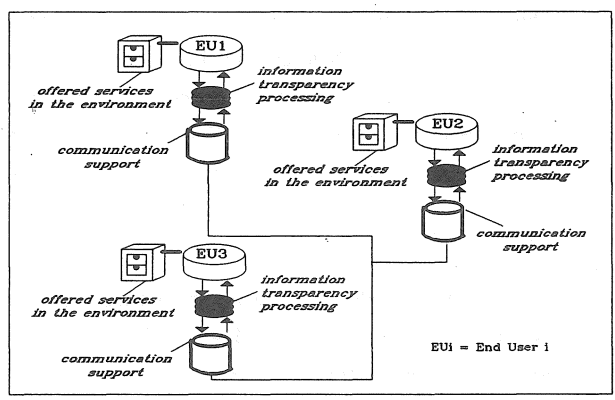

Fig 1: Interoperability factors

In relation to these three factors, we establish an overwiew of existing concepts and solutions that nowadays comply with interoperability requirements.

3. TAXONOMY OF INTEROPERABILITY

3.1. COMMUNICATION SUPPORT

The OSI Reference Model [ISO 7498-1], defines the seven-layered architecture from which are organized the open systems interconnection functions. The Physical, Data Link, Network and Transport layers focus on information communication, whereas the Session, Presentation et Application layers deal with information processing before this latter be really conveyed [3]. Nevertheless, the Session layer can be considered as the high part of the communication support since the virtual communication channel between two peer entities is managed in this layer. The Presentation and Application layers are more interested in data transparency and service accessibility factors.

Physical level

At the Physical level, interoperability seems well controled. The simple mention of the type of cable is often enough to define the entire physical appearance of the network

components and to specify what will connect and interoperate with what. This level of interoperability is made possible by efforts provided by organizations such as IEEE. The 802.3 (Ethernet-CSMA/CD), 802.4 (Token Bus) and 802.5 (Token Ring) LAN-intended standards specify exactly how should operate the physical layer of the network [4]

Data link level

In the Data Link layer, data is handled through formats which can differ according to underlying network architectures. Data link level interconnection solutions are **bridges**. This technology is used for extending a LAN to another LAN; it stores and forwards frames between networks. Most existing bridges require to operate on networks with similar or compatible architectures (eg: IEEE 802.x family) [5]. Networks with different architectures have to be interconnected at a higher level by other means, such as routers or gateways.

Network and Transport levels

At the Network and Transport levels, interoperability is based on semantic of bytes of a packet. In this context, the numerous Transport-layer definitions are not in favour of interoperability of systems. This plurality leads to a variety of record layouts and data fields meanings.

If a LAN has to be extended to a MAN or a WAN, the bridge technology is unsuitable. **Router** or even **gateway** technologies are necessary for interconnection of networks supported by different architectures. Routers [6] operate at the Network level. They identify the Network-layer protocols used by the two Network-entities connected, and ensure the routing for incoming packets

Every network architecture relies on particular protocols. These ones define the packet formats and the structures of handled addresses. The more usual are TCP/IP (Internet), XNS (Xerox), DECnet (DEC) and IPX/SPX (Novell). The existing router technology allows interconnection of network architectures unified around few de facto standards such as those mentioned above. In other case, specific gateways are required.

Session level

The Session layer main funtionality is to provide tools for dialogue management and synchronization [7]. Connection-oriented protocol and service have been specified for this layer by ISO [ISO 8326-8327]; a connectionless solution is currently studied [ISO DIS 9548-DAD3 8326].

The existing communication architectures show that the use of the Session layer is less obvious than the use of the further down layers. Numerous applications directly call for communication at the Transport-layer level. X-WINDOW, for example, is one of those

applications [8]. This is possible thanks to access interfaces such as Sockets or Transport Layer Interface [9].

The session concept is more concrete in the Systems Application Architecture (SAA) defined by IBM, thanks to its distributed transaction management solution: the Advanced Program to Program Communication (APPC) protocol specification and its implementation named LU6.2 [10]. Although this solution implements concepts specified in TP, the ISO application service element which deals with transaction processing [ISO 10026], it operates in an homogeneous environment built around LU6.2. Nowadays, the NetBIOS interface [11] is the best open solution corresponding to the OSI Session-layer. This API provides a set of services that allow a reliable full-duplex Session-level connection between two remote applications. NetBIOS is a de facto standard widely proposed upon LAN architectures.

3.2. TRANSPARENCY OF EXCHANGED INFORMATION

The second interoperability level concerns the exchanged message comprehension. Communicating entities must be able to receive or to send any data in a transparent and sensible way. An information format must be an obstacle neither for information communication nor for information interpreting. Therefore, the problem of various information representations that are met in an heterogeneous environment should be solved. This is the main function of the OSI Presentation-layer.

The OSI philosophy is the following: locally (i.e non-OSI world) the application processes use **concrete syntaxes** completely dependent on the hardware and software environment on which they run. Such syntaxes become unsuitable in an heterogeneous context, when remote processes have to communicate. Then, ISO proposes to hide the dependence between processes and their native environments. First of all, any couple of processes that wish to communicate should negociate one (or more) Presentation context. Such a context is defined by an **abstract syntax** (notation by which the process describes data units to be exchanged) and a **transfer syntax** (set of encoding rules which establish a common communication syntax for data units transmission) [7].

Both, a connection-oriented protocol [ISO 8822,8823] and a non oriented-connection protocol [ISO DP 9576] are specified for the Presentation level. In practice, de facto standard protocols that don't completely implement the OSI philosophy are proposed: XDR (SUN), Xerox Courier (Xerox) and NDR (Apollo/HP). They define coding rules for information representation (transfer syntaxes).

From an end-user viewpoint, the real benefits of interoperability result from the ability to deliver data, whatever their location may be, to anybody who need these data. Thus, the problem of information transparency joins the accessibility one.

3.3. ACCESSIBILITY TO OFFERED SERVICES

Interoperability is not only a sensible and synchronized information exchange on a reliable communication support. It includes capabilities thay any communication candidate entity should be aware of the offered and available services in the environment and the existing means to access them.

DCE [12], the OSF development platform for distributed environment, has been designed to provide both systems and applications interoperability. Among the DCE services, those relating to **communication** and **naming** offer solutions to problems of service accessibility.

OSF selected NCS 2.0 **RPCs** (DEC an HP) as the component for the DCE communication services [13]. RPCs aims at minimizing programming complexity relating to remote information transfer. This technology provides an identical behaviour regardless of the underlying transport protocol and ensures communication mechanisms transparency. RPCs mechanisms and ROSE [ISO 9072] have a similar scope: ROSE specifies an interaction support for remote communicating entities in conformance with the Client/Server model [ISO 10031-1], and RPCs are an implementation of this specification.

Naming services are particularly helpful in a distributed context. They allow objects (computers, files, etc) to be addressed regardless of their physical locations, by the use of user-oriented names which hide the access path complexity. Because of performance criteria, OSF has selected a local naming service (implemented on every cell of the distributed environment), and a global directory service for the whole environment. An application program uses the same interface to access either local or remote names. DIR-X (Siemens) is the remote naming service selected by OSF and is X500-compliant, the ISO directory service [ISO 9594]. For local names, OSF selected the cell naming service DECdns (DEC). The access interface to names is XDS (X/Open).

DCE provides accurate and concrete solutions which are helpul to solve problems of both applications and systems interoperability and supplies a global solution to distributed application programmers.

Our concern is to strengthen the accessibility aspect of interoperability with a ***service-oriented management system*** which manages end-user requests. This system aims at improving the use of offered services in the communication environment

4. SERVICE-ORIENTED MANAGEMENT SYSTEM MODELLING

Services are part of main actors that allow any organization to operate. Thus, their management is essential in order to allow an organization to operate in a consistent and suitable way. This management should ensure both **interoperability** and **quality of cooperation** within a network environment.

(Fig. 2) shows the proposed service-oriented system model. A ***service server*** allows end-users to be aware of all accessible services in the communication environment. Moreover, when several instances of a same service class are accessible in the environment, the server may select the instance the more convenient to an end-user requirement. It also ensures effectiveness of the offered service profiles.

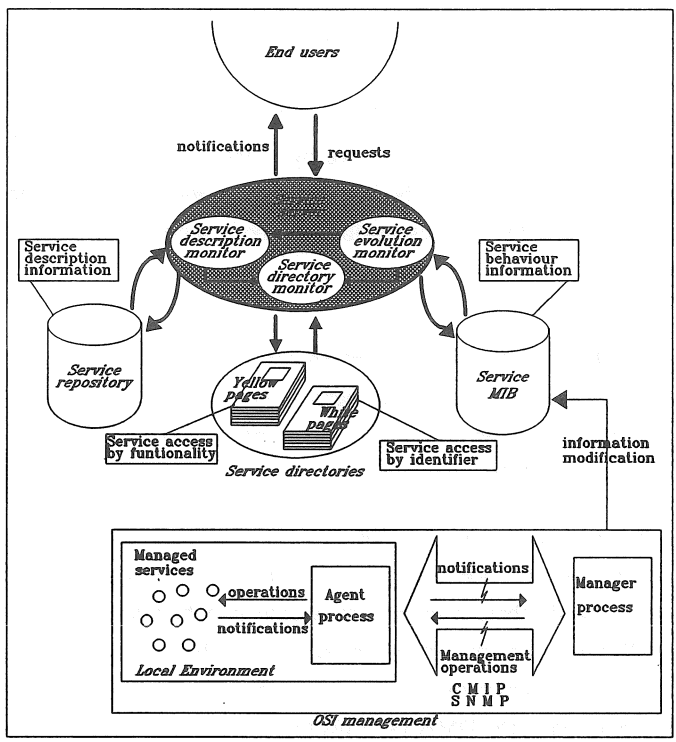

Fig 2: Service-oriented system modelling overview

4.1. WHAT ARE THE SYSTEM COMPONENTS ?

The system components may be classified within two groups: internal components and satellite components. ***Internal components*** define the management system kernel and perform all management functionalities. ***Satellite components*** supply useful information to internal components (Fig 2).

4.1.1. Satellite components

Satellite components of the service-oriented system are information storage tanks that allow internal components to operate. Three satellite components are identified:

The ***service repository*** contains all ***identification attributes*** (that describe a service in a static way) of monitored services,

The ***service MIB*** collects all ***status attributes*** (that describe a service in a dynamic way) related to service behaviour and performance.

Our concern can rely on the OSI management model, the basic principles of which are set in [ISO 10040], [ISO 10164-1..7] and [ISO 7498-4]. OSI management aims at providing mechanisms for interactive operating, monitoring and coordinating of resources. It defines solutions that ensure both management information access and transport [ISO 9595-9596]. The OSI management application relies on a management information base (MIB) [ISO 7498-4] which is a management object collection relevant to communication resources. Those objects are data supplied by the OSI management application components.

Service status attributes may be provided by a management application compliant with the OSI management model. Thus, beside the service-oriented management system, we consider an application which is responsible for both service activities observation and service MIB updating.

The ***service directories*** gather ***key attributes*** and allow to find a service location, either from its identifier (white pages principle), or from its functionality (yellow pages principles). This third component is useful to manage relationships between the service repository and the service MIB.

Convenient implementation solutions to these storage components are available. Databases may be used for identification and status attributes. More precisely, object-oriented and relational database management systems may be respectively used according to the complexity degree of structure object representation **[14]**.

X500 principles on which rely service directories may be achieved by using both DIR-X and DECdns included in the DME/DCE platform.

4.1.2. Internal components

Internal components define the service-oriented management system engine and are information customers. The highest level component is the ***service server*** we have mentioned abovel. It consists of the following sub_components:

The *service evolution monitor* consults and analyzes service management information (status attributes) collected in the service MIB in order to determine the required service availability, and eventually to select the most suitable service occurrence. Such a decision may be influenced by service location, service processing load rate, service failure statistics, and so on.

The *service description monitor* is responsible for management (consulting and updating) of service description information (identification attributes) gathered in the service repository.

The *service directory monitor* aims at refering and maintaining the service directories provided by naming service (yellow and white pages). More precisely, this monitor manages the service key attributes (service identifier/service functionality and service identifier/service location matchings). Functionalities of this monitor are used by both service evolution and service description monitors, when these latters have to search for a particular service.

4.2. WHAT ARE INTERACTIONS BETWEEN COMPONENTS ?

Interactions between internal components have to be completely defined. Satellite components rely on external tools to be used. An interaction between two components A and B is either considered as:

- a service request that A sends to B (if B is a processing component),
- an action that A performs on B (if B is a storage component),
- or a notification information that A sends to B.

Table 1 summarizes possible interactions which can be grouped in the three following classes:

4.2.1. Interactions between internal components themselves

The *service server* invokes services of the three monitors, according to end-user request semantics:

A consultation request aims at obtaining awareness of existing services in the environment. It requires *description monitor services*. These services may also be required for identification attributes updating.

An access request to a specified service requires *evolution monitor services* in order to both verify service availability and perform service selection if necessary.

Directory monitor services may be used in order to satisfy an updating request of key attributes stored in directories (identifiers, locations, functionalities).

Interaction issued from the service evolution monitor

The *service evolution monitor* uses *description monitor services* to retrieve complementary descriptive information about a service, in order to respond in a significant way to a service server request (eg: searching for a service selected authority).

Moreover, *directory monitor services* can be requested to provide a service real location address from this service identifier, or to obtain a service identifier from the service functionality.

The *service server* is notified with operation results obtained by the evolution monitor.

Interaction issued from the service description monitor

The *service description monitor* requests *directory monitor services* to quickly locate a service according to its functionality. It notifies the *service server* with obtained operation results or the *evolution monitor* with required service identification information.

Interaction issued from the service directory monitor

The *directory monitor services* sends service identification information to the *evolution monitor* or *description monitor*.

4.2.2. Interactions between internal components and satellite components

The *service evolution monitor* interacts with the *service MIB* by collecting service status attribute information to analyze required service availability and performance.

The *service description monitor* interacts with the *service repository* to provide service identification attribute information or update this latter when it is requested by the service server.

The *service directory monitor* interacts with the *service directories* to supply or modify a service identifier, a service location or a service functionality.

4.2.3. Interactions between internal components and end-users

End-users access the *service server* by service requests. Services server parses these requests and interact with end-users to notify them of a service result, a service failure or other notifications. The service server is the access point to the service-oriented management system.

INTERNAL COMPONENT	*INTERACTS WITH*
Service server	Service evolution monitor Service description monitor Service directory monitor End-users
Service evolution monitor	Service description monitor Service directory monitor Service MIB
Service description monitor	Service directory monitor Service repository
Service directory monitor	Service directories

Table 1: Internal component interactions

5. PERSPECTIVES: PERFORMANCE EVALUATION AND IMPLEMENTATION

The proposed service-oriented system aims at both providing end-users with information about accessible and available services and ensuring quality of provided services. We have exposed in this paper the basic architectural principles of our solution, and a general specification of the components.

We currently focus on a formal internal components modelling. Interface description tools such as IDL language [15] and simulation tools such as QNAP2 [16] help us to get a well defined component interfaces and to evaluate implementation performance. Thus, we want study the feasibility of the proposed model in terms of performance, particularly relating to response time and to the throughput introduced on the network.

Implementation perspectives lead us to consider a de facto standard open system, OSF/1 [17], which may support a development and management environment for distributed applications: OSF DCE/DME [12][18]. DCE offers solutions for both application and system interoperability. As soon as it is available we will implement processes that support the service-oriented system in this environment.

BIBLIOGRAPHY

[1] G.J. NUTT, **Open systems : integration, interoperability, portability,** Prentice Hall, 1991

[2] ISO/IEC JTC1/SC21 N4884, **Basic Reference Model of Open Distributed Processing,** July 1990

[3] A. TANENBAUM, **Computer networks,**Second edition, Prentice Hall, 1989

[4] P. ROLIN, **Réseaux locaux : normes et protocoles,** 3e Ed., HERMES, 1990

[5] M.A. MILLER, **Internetworking,** Prentice Hall 1991, A guide to network communications - LAN to LAN; LAN to WAN

[6] J.M. Mc QUILLAN, E.J. TEJA, Data Communication, September 29 1989, **Routers as building blocks for robust internetworks**
Router roundup : tools for network segmentation come of age

[7] H. NUSSBAUMER, **Teleinformatique III, Session, Presentation Compression de données, Couche d'Application,** Presses polytechniques et universitaires romandes, 1991

[8] T. DESPRATS, A. BENZEKRI, C. ARMENGAUD, **Ouverture des systèmes multifenêtre et intégration de technologies multimédia: situation et perspectives de normalisation,** Actes Convention Informatique Latine 91, Barcelone(Boixareu editores), Juin 1991

[9] W. R. STEVENS, **Unix network programming,** Prentice Hall, 1990

[10] J.P. DURAND, **Etude et réalisation de communications sous APPC/LU6.2 pour la mise en oeuvre de traitements coopératifs,** rapport interne SIERA, Juin 1991,

[11] B. GLASS, **Understanding NetBIOS,** Byte, January 1989, volume 14, number 1

[12] **Distributed Computing Environment Rationale,**Distributing Computing Environment Evaluation Team, Open System Foundation, 14 May 1990,

[13] B.HIRSCH, OSF DCE RPC Evaluation Report, Eureka Software Factory, 1991, **RPC Evaluation Criteria : a checklist of RPC**

[14] M. SIBILLA, F. FAURE, T. DESPRATS, A. VALDERRUTEN, **Information base modelling for open heterogeneous network management,** IFIP/IEEE Workshop on Distributed Systems : operation and management, DSOM'91, Santa Barbara, California, U.S.A., October 15-16, 1991

[15] **The Common Object Request Broker: Architecture and Specification**
OMG Document Number 91.12.1, Revision1.1, Draft 10 December 1991

[16] Bull, INRIA (1984) & SIMULOG (1990).

[17] **Guide to OSF/1 : a Technical Synopsis,** O'Reilly & Associates Inc, 1991

[18] M. AUTRATA, **OSF Distributed Management Environment,** IFIP/IEEE Workshop on Distributed System : operation and management, DSOM'91, Santa Barbara, California, U.S.A., October 15-16, 1991

[19] **HP OSF/1 Technology Release System,** Documentation Hewlett-Packard 1991, Programmer Guide, Programming Environment Reference, NCS 2.0 Programmer's Guide
NCS 2.0 Programmer's Reference

ABBREVIATIONS

DCE	Distributed Communication Environment
DME	Distributed Management Environment
IDL	Interface Definition Language
ISO	International Standard Organization
LAN	Local Area Network
MAN	Metropolitan Area Network
MIB	Management Information Base
NCS	Network Computing System
ODP	Open Distributed Processing
OMG	Object Management Group
OSF	Open System Foundation
OSI	Open System Interconnection
QNAP2	Queueing Network Analysis Package 2
ROSE	Remote Operatiion Service Element
RPC	Remote Procedure Call
SAA	Systems Application Architecture
TLI	Transport Layer Interface
WAN	Wide Area Netwok
XDS	X/Open Directory Service