Specification of Communication Services for Multimedia Cooperative Applications Based on the Reference Model of ODP

Jong-Hwa Yi and Encarna Pastor

CONTENTS

ABSTRACT

Between a wide range of distributed applications, this work is particularly focused on those applications which allow a group of persons, who are geographically distributed, to interact and cooperate exchanging different kind of information in order to realize group activities in a common work environment. In this paper, we realized a study of this kind of applications to identify general and functional characteristics which will then enable us to determine their communication requirements. Also, we analyzed the existent support architectures to examine whether they support multimedia cooperative applications. Based on the results of analysis, we proposed a set of communication functions needed additionally to support these applications, using the ODP(Open Distributed Processing) modelling concepts, rules and viewpoints as our global design methodology.

I. INTRODUCTION

Distributed application development needs support environments or platforms which offer a set of facilities to cover the different requirements of various types of applications. Evidently, if the development is to make use of more advanced technology, for example, the creation of a common cooperative work environment for a group of persons, the use of distinct types of information such as text, graphic image, audio and video, etc., this support must be even more complete and flexible. This trend means that such environments must not only understand the different application needs but also be capable of adapting to rapidly changing user needs and be able to take advantage of the benefits of technological evolution.

Within the field of distributed processing, the possible application areas are almost unlimited, since there are no limitations on the creation of new application in accordance with the specific organization or user needs. Since the functional requirements of distributed applications are varied and very different, truly useful support architectures must be capable of providing a set of suitable facilities to cover their requirements. In this way, the applications need only to express the level of support required and do not need to concerned with how this support is to be obtained.

Between a wide of variety of distributed applications, in this work, we consider particularly those applications which support the cooperative activities between a group of persons and additionally want to use different kind of information such as text, graphic image, audio and video. The applications which demand these characteristics have been studied in CSCW (Computer Supported Cooperative Work) area. The objective of CSCW is to solve the needs and problems presented when a group of persons wants to work or cooperate between them. In CSCW area, we investigate exclusively on what technological solutions are needed for the development of cooperative applications in the underlying support architecture. We consider the standards for Open Distributed Processing (ODP) as an appropriate architecture which can support the development of cooperative applications. From the CSCW viewpoint, this kind of applications can obtain open solutions for their technological support based on the generic architecture proposed by ODP. From the ODP viewpoint, its architecture can understand requirements of new field of applications, in this case multimedia cooperative applications, and include additionally a number of useful facilities in its architecture. In this mode, the scope of ODP standardization will be extended with aim to support a wide range of distributed applications.

The standardization organizations ISO and ITU-T have cooperated in the definition of the Reference Model of ODP (RM-ODP). As mentioned above, the objective of ODP is to define a common architecture which supports the specification and development of systems and applications in distributed and heterogeneous environments. In parallel of the standardization effort of these organizations, several universities and industrial organizations have carried out numerous research projects, from which have resulted *de facto* standards and support the implemented platforms which help the application development.

On the other hand, in CSCW area there is an increasing demand for research and development of collaborative applications. One of the projects in which we participated, concretely in education field, is the CO-LEARN (Collaborative Learning) project in the DELTA Programme of the European Commission. It started in 1992 and had a duration of three years. The main aim of this project was to develop an operational telematic environment which can support educational collaboration between users such as tutors, teachers, colleagues at work and learners [36], where our task was focused on the provision of communication services needed to support different kind of collaborative interaction.

When we were working on this project, some of the main difficulties we faced were the lack of a generic platform which offers the interoperability and portability of the developed applications, and the lack of stable technologies to support different types of group interaction. These technical reasons have represented an important motivation to initiate a theoretical investigation on communication support for cooperative applications in a framework proposed by the reference model for ODP which guarantees the interoperability and portability, and provides a set of necessary functions to build distributed applications.

This paper is organized as follows. Firstly, we analyzed a set of general characteristics of multimedia cooperative applications in order to determine their communication requirements which must be covered by a support architecture. Secondly, we examined various existent architectures to evaluate whether they support the communication requirements identified in the previous step. Finally, based on the results of this evaluation, we specified a series of communication functions needed to support the development of multimedia cooperative applications in ODP environment.

II. ANALYSIS OF MULTIMEDIA COOPERATIVE APPLICATIONS

We have studied in detail and from a general application view point on different application areas with the aim of understanding its functional and technological requirements which should be covered by the support environment [3]. In order to accomplish this, we analyze general characteristics arising in the field of multimedia cooperative applications and then identify communication requirements derived from

Table 1. Classification of CSCW Applications.

		Collaboration Time	
		same time	different time
Geographical Distribution	local	room meeting	group scheduler, white board, *etc.*
	remote	teleconferencing cooperative learning, teleworking, *etc.*	mailing, asynchronous multiuser editing, *etc.*

1. General Characteristics

Between a wide range of distributed applications, we consider in particular those applications which intent to solve the needs and problems presented when a group of persons wants to work or collaborate between them, and which wants to use different types of information. Considering this kind of applications which currently have significant importance and which have attractive characteristics both from the general user view point as well as from the view point of this work, the following characteristics and requirements were identified:

- **Distribution and Collaboration**: In the literature there are various concepts and taxonomies used in the classification of CSCW applications [7], [14]. One of the most general definitions is based on the geographical distribution of the participants and the forms of interaction between them. According to this definition, the group members can participate with different mode, depending on the member work site (local or remote) and on the cooperation time (synchronous or asynchronous mode).

  Table 1 shows the classification of CSCW applications in four categories indicating some possible applications. This classification has two important consequences. Firstly, these applications need to support different types of interactions/relationships that a group of individuals establishes in order to carry out certain cooperative activities [5]. Secondly, the most work environments of the group members are typically distributed and therefore the members must cooperate through the communication networks to which they are connected. In the latter case, the communication support must have the capability to offer different forms of interaction, as well as the capability to offer useful services with the guarantee of some required level of quality.

- **Use of different types of information**: This is one of the requirements of those users and organizations are currently showing the most interest for their work environments. Different types of information used in a distributed application impose strong requirements with respect to the support systems, leading to the need for different levels of support technology such as storage capacity, synchronization mechanisms and communication capacity [9], [31].
  With respect to the communication capacity, continuous media such as audio and video require principally the use of high speed/performance networks. These networks must be capable of supporting sufficient bandwidths for the transmission rates required by each media type and quality of services expressed by the application which use the continuous media [32].
- **Support for high interaction**: Each application could be needed a different grade of interactions among its users, however, we can also consider those applications which require operations or interactions in a relatively short space of time, such as applications which support highly-interactive cooperative activities, which handle continuous media or systems/applications for one-line transactions. Consequently, a method which permits the specification of requirements in function of the quality of service is needed and for this purpose the QoS parameters are normally used [7], [29], [30].
- **Distribution Transparencies**: In a distributed environment, there is a need for mechanisms which hide distribution properties, in order to the application designers only need to express the kind of transparency required, without worrying about how to obtain these transparencies. The distribution transparencies may be selective [1], [2], [10]. It means that the designer can select the grade and type of necessary transparencies depending on the application, since a given application will usually require only some of the distribution aspects rather than all of them.

2. Communication Requirements

The different application requirements should be covered in a transparent manner by a collection of facilities offered by the support environment, that is, by ODP systems which guarantee the provision of the support functions at different levels such as at the level of architecture, storage, processing, communication protocols and networks, development tools, etc. [1], [3], [7]. Among these, we concentrate exclusively on analyzing the communication support.

In the following we summarize the different types of communication requirements which are derived from the general characteristics identified in the previous section.

1) **Support for Group Interaction**: Different types of communication services enabling distinct forms of inter-

action (synchronous or asynchronous) and interaction styles (between members within a group, one person to group, group to group) are required. If the interaction styles are one person to group or group to group, a group invocation mechanism could be used [12]. The group invocation involves only one operation which is carried out simultaneously on a collection of group members. In principle, there is no difference between the group invocation scheme and the individual invocation scheme. A group invocation may have either the interrogation or the announcement semantics of the client and server model.

2) **Group Members Management**: A group is formed by various members and their behaviors are dynamic. It means that a new member wishes to join the group, a current member wants to leave the group, a member can be registered in various groups, a member can participate in different group interactions at the same time, etc. [6]. For this reason, the dynamic management of the group, which verifies that the members behave correctly according to the established rules and policies of the group, is needed.

3) **Group Communication**: Since the interactions involve a set of group members, group communication support is required. Such support permits the connection establishment between different members with different styles (1:1, 1:M, N:1 and N:M) and facilitates the simultaneous exchange of information between them. In this case, the information exchange should provide different transmission mode such as unicast, multicast and broadcast. One important function in group communication is the dynamic connection management, since the behavior of the group members can be changed dynamically where a new connection must be established or a current one must be released continuously [15].

4) **QoS Guarantees**: From the application view point, it is not sufficient to provide the services required, it is also necessary to guarantee certain levels of service quality required by the application. For this reason, this is one of the most important requirements to demand of the communication-level support [9], [29], [30]. With respect to this requirement, a set of functions is needed to respond to quality demands of applications.

- **Specification**. A set of flexible QoS parameters should be defined to that each application can express its level of quality service needed using these parameters, for examples, throughput, end to end delay, transmission delay, error rate, priority, jitter, latency, etc.
- **Negotiation**. The level of QoS spec-

ified by the application must be negotiated between parties which will be connected. This negotiation occurs in the moment of the establishment of connection and the values agreed between them will be maintained from the connection establishment to the connection release.

- **Renegotiation**. In some cases the application needs to change the previously defined QoS level. For this, a function which enables the application to modify the condition of QoS is needed during the connection is active.
- **Monitoring and Notification**. These functions are to check whether the level of QoS established is maintained or not, to notify some errors occurred in communication subsystem or to notify the modification of the QoS parameters and its values.

5) **Multimedia data transmission**: Audio and video data are continuous media. This is since these media can be considered as a series of frames of finite size which are generated, transmitted and received in fixed time intervals, so that these time intervals must never be exceeded [33]. With the aim of supporting the transmission of multimedia data, the provision of a flexible range of qualities of service is important.

III. EVALUATION OF DISTRIBUTED ARCHITECTURES

Architectures that provide a framework for the specification of systems and applications and some implemented platforms which facilitate the distributed application development are available. However, these architectures and platforms have been independently defined in accordance with policies and interests, and using the in-house technology of each of the industrial organizations in which they originated. In consequence, in addition to the ODP Reference Model carried out by ISO and ITU-T, numerous *de facto* standards such as ANSA, TINA, CORBA exist in the distributed processing area [1], [2], [3], [16], [22], [28].

In recognition of the different communication requirements of applications identified in the previous section, the objective of this section consists of the evaluation whether the currently available support architectures satisfy them, indicating where they meet these requirements and their limitations with respect to them.

1. Reference Model of ODP

Thanks to the standardization effort carried out by the standardization organizations, ISO and ITU-T, the *de jure* standard is available, the RM-ODP, which defines a consistent and common support architecture. It introduces viewpoints for the design of distributed systems and applications, each with an associated viewpoint language which expresses concepts and rules relevant to a particular set of concerns [1], [8].

To support different interaction types between objects, the computational model defines several interface types: *operational, stream* and *signal* interfaces. It indicates that application objects which want to use audio or video data can specify stream interfaces to handle or interchange this kind of information. All interactions between computational objects are only possible when their interfaces are bound. Bindings between objects may be established with or without a *binding object* which provides links between computational objects with a set of communication support functionality. Each application can configure a binding object depending on its requirements such as link type, number of objects which participate in the interaction, information types, QoS, etc. In this model, the compound binding is also introduced for interactions between more than two objects where a group interaction of objects can be specified using this concept [2].

The engineering model specifies services and mechanisms necessary to support the computational model. In this model, various engineering objects which support distribution functions required in a distributed environment are defined. One of them is *channel* which supports distribution-transparent communication services and mechanism between objects. Different kinds of channel can be configured to support a specific interaction type defined in a binding object in the computational model. The RM-ODP considers three channel types to be fundamental: a channel for operation execution between a client and server object (operational), a channel for group interaction and a channel for stream interaction [4].

In general, the RM-ODP considers the issues of the group interactions and the handling of different types of information as previously mentioned. However, this reference model does not address the abstraction concepts needed to satisfactorily model these issues. Some extensions for supporting group interactions and the handling of streams are studied and proposed [1], [4], but these extensions proposed do not completed to respond to the communication requirements analyzed in the previous section.

2. ANSA

ANSA (Advanced Networked System Architecture) originated in a project within the UK Alvey Information Technology Programme, after that APM (Architecture Projects Management Ltd.) took over the ANSA work. It is a generic architecture which supports the design and implementation of systems and applications in distributed and heterogeneous environments [10], [13]. The ANSA work was served as a significant input for the development of the RM-ODP, so these two architectures share the basic concepts and architectural principles.

The concept of *group* introduced in ANSA is a well-known concept used in a wide variety of distributed systems in order to obtain improved service availability, efficiency through parallelism, fault tolerance, etc. In this context, a group is formed by a set of server objects, each one performs exactly the same service in parallel. A group is defined as an object with two interfaces: *group service interface* and *group control interface*. An invocation on the group's service interface will be carried out on all the interfaces of the active group members. The group control interface can be used to define operations to manage and configure the group behavior [11]. In this context, this concept can be used to present a group which is formed by a number of participants. Each participant can send to or receive from messages via a group service interface and the behavior of each participant could be controlled via a control interface. However, in this case, each participant is presented as an object which has dynamic behavior. It means that the dynamic management of participants is important issue to consider for the group interaction, because a new participant wishes to join the group, a current participant wants to leave the group, a participant can be registered in various groups at the same time, etc [15].

In addition of the REX (Remote Execution) protocol, ANSA offers the GEX (Group Execution) protocol to support group communication. If a client invokes a group interface, GEX creates a group session, establishing point-to-point connections between a number of servers using the REX protocol.

In order to support applications which handle continuous media types, some extensions have been made to ANSA and its platform named ANSAware [34]. The approach taken in these extensions for multimedia is to add functionality in the form of low level multimedia base services, such as streams connections between a set of media sources and a set of media sinks, multimedia devices and chains, and a group of sources endpoints and a group of sink endpoints. The current APM research activities consist on the design and development of new concepts and mechanisms to support a generic distributed real-time computing environment, investigating strongly real-time requirements driven by continuous media such as audio and video data [35].

3. TINA

Telecommunications Information Networking Architecture (TINA) is a reference model based on the RM-ODP which covers the telecommunication domain [22]. This architecture defines a Distributed Processing Environment (DPE); a homogeneous infrastructure which marks the complexity and heterogeneity existent in sub-networks and distributed resources [24].

Like the RM-ODP, TINA defines five viewpoints adopting the same concepts and specification languages for distributed systems and applications. The interaction model of this architecture introduces distinct interface types. The operational interfaces in which objects interact by invoking operations and sending responses, and the stream interface in which objects interact by exchanging stream flows are defined [23]. Also the concepts of binding object in the computational model and of channel in the engineering model are included. As previously mentioned in Section 1, some new concepts and functionality must be extended to support group interaction and stream handling. Some detailed issues that TINA Consortium has identified as further studies are the presentation of characteristics of continuous media and the invocation method in the stream interface, specification of binding objects for group communication, functionality of group and stream channels, etc.

In addition to the global architecture TINA, four sub-architectures: *Service Architecture, Network Resource Architecture, Management Architecture* and *Computing Architecture* are specified. Among them, the first and second architectures define a set of useful objects which offer session or communication services needed to develop telecommunication applications. In particular, the purpose of SSM (Session Service Manager) and CSM (Communication Service Manager) objects is to maintain and control established sessions, and to provide a set of communication services for that application objects can interact exchanging stream flows between them [25]. CSM is considered as one class of binding objects specified by TINA architecture to support group and stream interaction. However, the specification of binding objects and channels which offer suitable services and mechanisms to cover a variety of application communication requirements must be studied and proposed.

4. CORBA

The Object Management Group (OMG), which is an industry consortium dedicated to creating standards in object-oriented environments, has specified Common Object Request Broker Architecture (CORBA) which is composed of four main constituents: *Application Objects, Common Facilities, Object Request Broker* and *Object Services* [16], [17], [18], [27]. CORBA is an architecture which provides a minimum set of services and tools for the distributed application development.

The standardization activities of the OMG which have carried out so far deal mainly with the overall architecture, the *Object Request Broker* and the *Object Services.* In consequence, there are already standards published in this area and the current standardization is focused on the *Common Facilities*, which is the level that most application designers will use. From the general application of view, the services actually defined in the *Common Facilities* are lower-level services and are not sufficient to cover the majority of distributed application requirements, and in particular those requirements are discussed in this paper are not covered. It is because CORBA is focused on, in particular, application areas such as finance, medicine, manufacturing, etc. However, we expect that other support services which are additionally needed for other distributed applications can be added and implemented on this global architecture. Also, the OMG group generated various Requests For Information (RFIs) and Requests For Proposals (RFPs) for the CORBA components in order to study and analyze new services, mainly in areas where industry consensus already exists. In this mode, this group can extend their standardization scope and in consequence CORBA can offer more useful and flexible services requested by different application areas, including multimedia cooperative applications.

To add a new service in the *Common Facilities*, it is important to determine which new system services are also needed in other architecture components in order to support it at the level of the overall architecture. In this context, we consider that to cover the communication requirements identified in the previous section, the facilities for group interaction including group creation with group property and group members management could be defined and included into the *Common Facilities*. The facilities for creating/destroying group interfaces and those group invocation methods could be included in the *Object Services* component. Capabilities such as those enabling group communications, continuous media exchange, multiconnection establishment and management with required QoS, etc., could be incorporated into the *Object Request Broker* component [15].

5. Conclusion of Evaluation

The detailed evaluation of the architectures mentioned leads to detect the lack of flexibility and required functionality to response to the communication requirements of multimedia cooperative applications. To completely support the communication requirements identified, we identified the following functions which must be included in a support architecture.

- *Dynamic Group Management* which is concerned with group configuration, group behavior management and membership control such as authentication, access control, role control, etc.
- *Group Stream Interaction* in order to provide communication services needed in supporting group interaction exchanging stream data. In particular, *Compatibility Checking for Stream Interfaces* and the *QoS level* in the establishment of binding between stream interfaces.
- *QoS Management* which enables application to specify QoS parameters, negotiate the initial level of QoS, renegotiate the previously-defined QoS level and notify some QoS modification or errors occurred.

- *Stream Interface Type Verification* which stores the information relevant to stream interfaces and checks the compatibility between two stream interfaces in accordance with the comparison rules.

In the next section we will propose a communication model which provides a set of services to cover the communication functions described above.

IV. SPECIFICATION OF COMMUNICATION SERVICES

In this section, we specify a set of communication functions which provide useful services to respond to the communication requirements identified previously. These communication functions must be included in a distributed support environment or system for that the application designers can use them to develop distributed applications.

The concepts, rules and different viewpoints defined in the Reference Model of ODP, *the Models of Information, Computation, Engineering* and *Technology*, are used to specify the communication services as our overall design methodology.

1. The Information Model

Using the Object-Oriented Analysis (OOA) methodology proposed by [21], Figure 1 illustrates the information model in which a set of information objects and their attributes, and the relationships between the objects are defined.

The simple description of the information objects is as follows.

- **Participant**: This object represents a participant or a group of participants which wants to enter into a session.
- **Group manager**: If a group of members participates in a session, this object is responsible for controlling the correct behavior of group members in accordance with the established group rules and policies.
- **Session**: Corresponds to a session in which the participant(s) can collaborate between them in different interaction mode. Also the participants can exchange information of distinct types.
- **Session manager**: This object manages all created sessions and also maintains up-to-date information concerning the currently active sessions.
- **Association**: The objective of this object consists of offering suitable communication services in order to support the group interactions, including the establishment of connection between the group members, facilities to send or receive multimedia data, dynamic control of the established connections, etc.
- **Information**: This object represents different types of information: Text, Graphic image, Audio and Video. In a session, the participants can use the type of information they want and the information will be exchanged via an as-

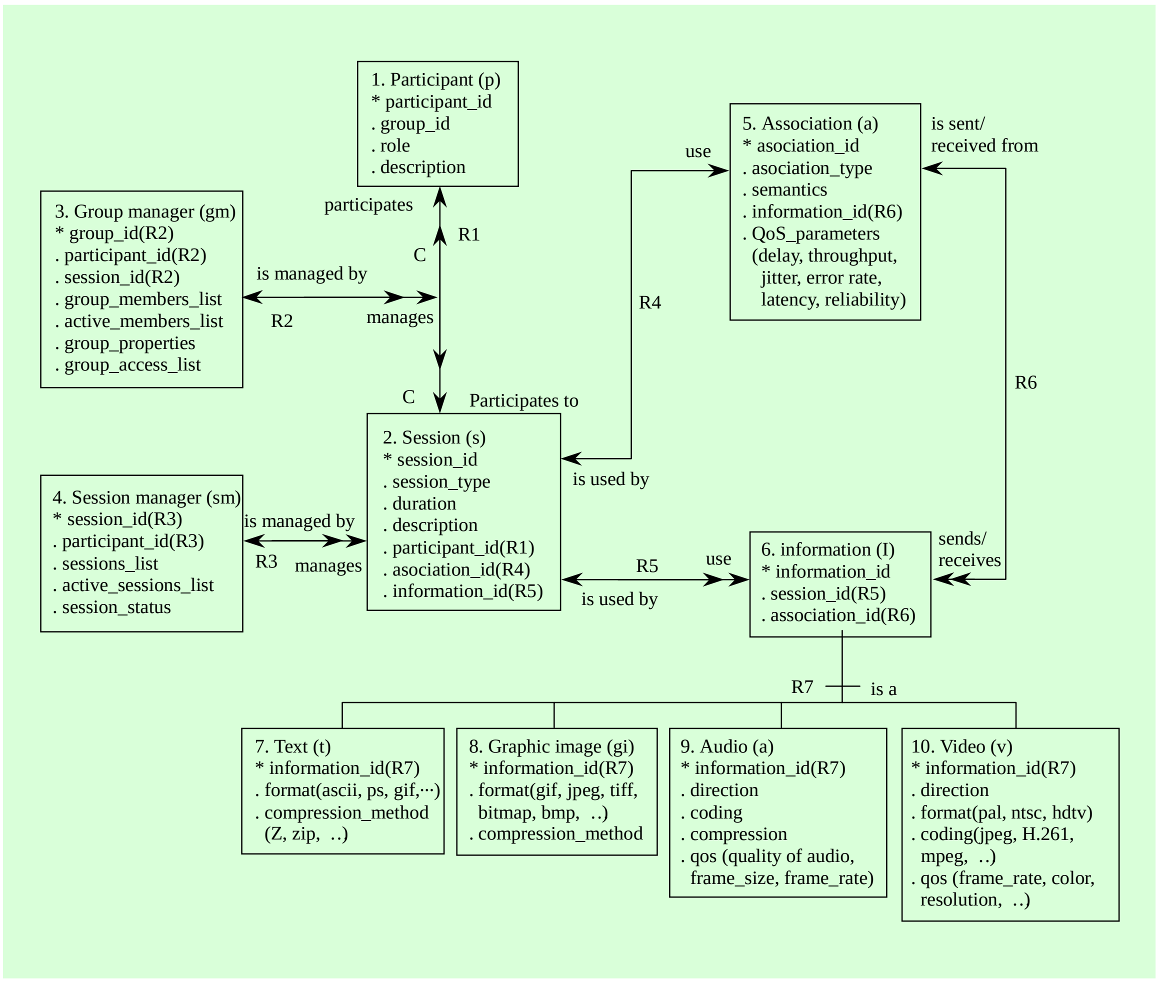

Fig. 1. Information model.

sociation connected to them.

2. The Computational Model

To provide the necessary communication services for the group interaction, in this model we defined the computational objects: *Group Manager, Type Manager and Group_Stream_Binding* (Fig. 2). The Participant is an application object which represents a participant and several participants form a group. The participants in a group interact via its Stream_Group_Binding object which offers group communication services. We use the Object Description Language (ODL) proposed by TINA-C [26] to specify the computational objects, their interfaces and operations.

1) Group Manager

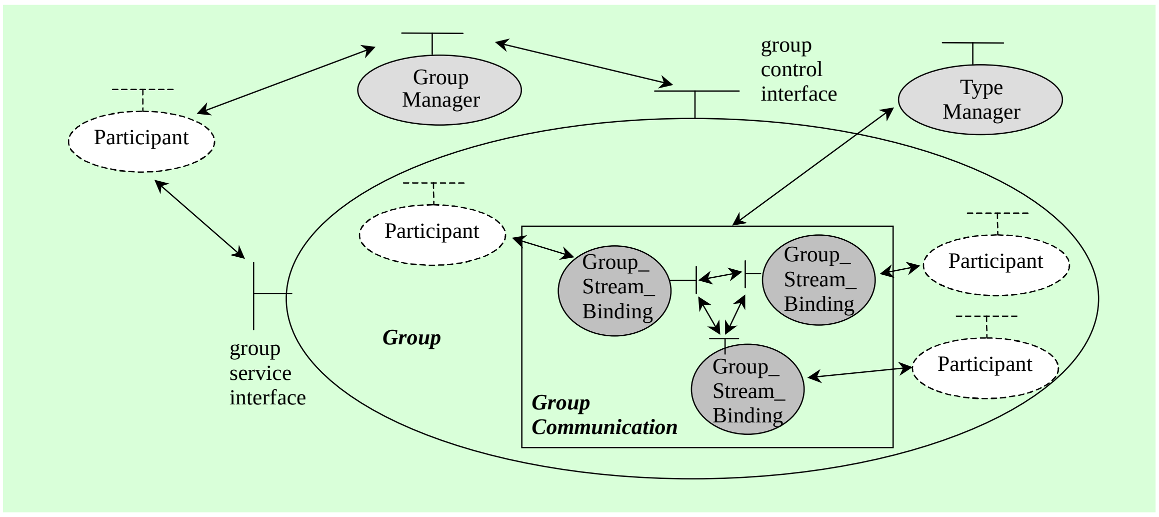

Fig. 2. Computational objects.

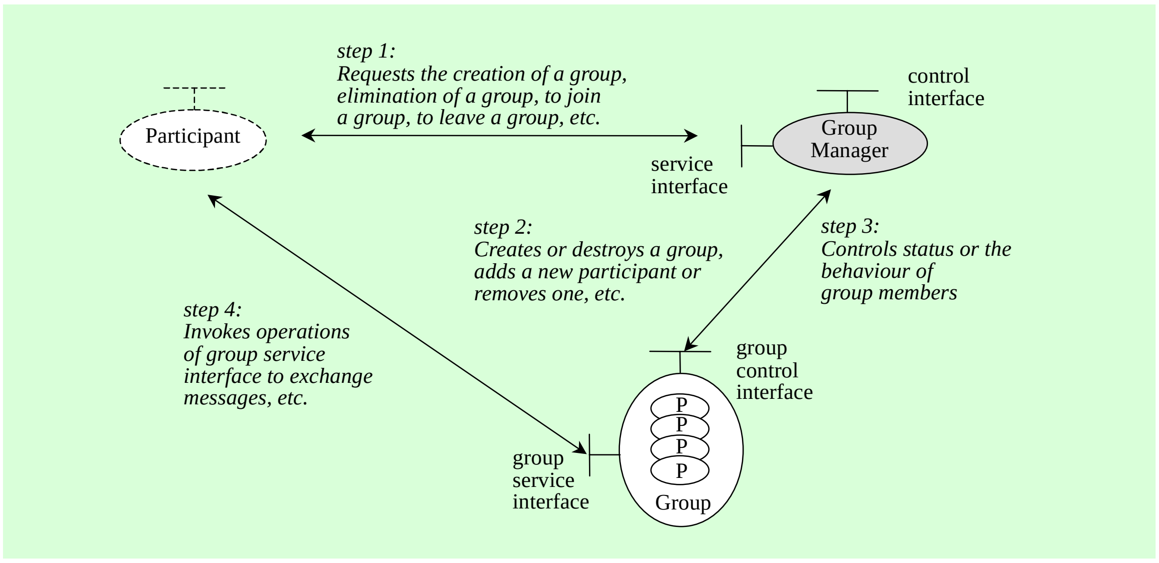
step 1: Requests the creation of a group, elimination of a group, to join a group, to leave a group, etc.
control interface
Participant
Group Manager
service interface
step 2: Creates or destroys a group, adds a new participant or removes one, etc.
step 3: Controls status or the behaviour of group members
step 4: Invokes operations of group service interface to exchange messages, etc.
group control interface
P
P
P
P
group service interface
Group

Fig. 3. Group manager.

The purpose of this object is to manage the group interactions (Fig. 3). Any application object can request the creation of a group, the elimination of an existing group or simply invoke other operations offered by the *Group Manager* object to get some information such as group properties, list of members, general group description, etc. (step 1). When receiving the request to create a new group, this object returns a

Table 2. Specification of the Group_Manager_service_interface.

```
interface template Group_Manager_service_interface {
operations
void Create_Group (in Group_id group, in Property_Group property);
void Destroy_Group (in Group_id group);
void Join (in Group_id group, in Paricipant_id participant, out boolean result);
void Leave (in Group_id group, in Paricipant_id participant);
void List_Groups (out Group_id group);
void List_Group_Members (in Group_id group, in Paricipant_id participant);
void List_Active_Members (in Group_id group, in Paricipant_id participant);
void List_Group_Property (in Group_id group, in Property_Group property);
behaviour
};
```

group service interface and a group control interface (step 2), where application objects can use the first interface to exchange messages between them (step 3) and the *Group Manager* can invoke the another interface to verify that the group members behave correctly according to the established rules and policies (step 4). For example, it checks whether each member that requests entry intothe group can be accepted or not and it checks the members access permissions as well as their assigned roles in the group. It maintains also up-to-date information concerning the currently active members. The Group Manager object has two interfaces to offer the services described in Table 2.

2) Group_Stream_Binding

This is a kind of binding objects which provides communication services to support interaction between group members and to exchange stream data. To accomplish this, this object provides several functions such as establishment and destruction of multiple bindings, binding management, guarantee of service quality, group interaction facilities, exchanging of stream data. For this, three different interfaces of this object are defined using ODL.

In the establishment of bindings capable of responding to the communication functionality required, several previous steps are needed. These steps consist basically of checking the compatibility of both interface types and of QoS levels. In the first case, it must be checked that both of the interfaces between which the bindings are to be created are stream interfaces, that the type of streams (audio, video or both) handled by each of these interfaces are compatible and that the direction (in, out or both) assigned to each stream of each interface is compatible. In the second case, the common conditions for the required level of service quality must be negotiated or, in the case where it is impossible to offer this required level, an agreement must be reached as to the most adequate level.

Table 3 shows the ODL specification of the interface *Group_Stream_Binding_Service_Interface* of this object.

3) Type Manager

The main function of this object is to maintain a set of information about stream interfaces such as interface type (name) of an object, interface classes (operational or stream), kind of streams of that interface (audio, video or both), direction of each stream type, etc. Based on the information stored, this object provides a set of operations which permits an object to add its interface type, to delete one existent or to request the compatibility check between two stream interfaces.

3. The Engineering Model

Two types of channel: *Operational Channel* and *Stream Channel* which support distribution-transparent interaction between the computational objects are defined in the engineering model.

Based on the channel concept of ODP, in this paper we present only the configuration of a ***Group_Stream_Channel*** in order to offer a set of communication services for the *Stream_Group_Binding* object defined in the computational model. The *Group_Stream_Channel* is composed by a set of objects: *Group_Stream_Stub, Group_Stream_Binder, Group_Stream_Protocol* and *Interceptor*. The specification of this channel and the control interface of the binder object is as shown in Table 4 and 5.

Figure 4 illustrates the configuration of this channel which offers communication services to support group interactions involving stream transmissions. For it, each object of the channel provides a set of functions through its service and control interface. In addition to the channel object, ***Nucleus*** object is specified. In principle, this object provides basic management functions for a node including some communication facilities such as the creation/destruction of a channel, the creation of a reference of channel, etc. A *Participant* object can solicit a channel (in this case *Group_Stream_Channel*) to *Nucleus*. Then Nucleus creates a requested channel and returns a channel reference to the Participant object.

4. The Technology Model

In this model, we just describe a possible development environment in a node which is devided into three component layers:

- **Application layer**: In this layer, there are application objects which will use the support functions provided by the *Support Platform*.

Table 3. Specification of the Group_Stream_Binding_service interface.

```
typedef struct {
        string productor_id;
        string consumer_id;
        string group_id;
} Identification;
typedef struct {
        string stream_name;
        enum direction {productor, consumer};
} Stream_Description;
typedef sequence <Stream_Description> Stream_Information;
typedef struct {
        string interface_name;
        enum interface_type {operational, stream};
        Stream_Information stream;
} Interface_Description;
typedef sequence <Interface_Description> Interface_Information;
typedef struct {
        enum mode {unicast, multicast, broadcast};
        string destination_users;
} Send_Mode;

interface template Group_Stream_Binding_Service_Interface {
operations
void Establish_Group_Stream_Binding (in Identification user_id, in Interface_Information interface,
            out int bind_id) with QoS_Data qos;
void Destroy_Group_Stream_Binding (in Identification user_id);
void Send (in int bind_id, in Identification user_id, in Send_Mode mode, in MM_Data_Description
            description, in char* data);
void Receive (out char* buffer);
void Start (in int bind_id, in Identification user_id, in Send_Mode mode, in MM_Data_Description
            description, in char* data);
void Prepare();
void Stop (in int bind_id);
};
```

- **Support Platform layer**: This layer contains an infrastructure required to support open distributed processing environment, including a set of functions needed to offer distribution transparencies which make the designers possible to implement applications. Also, this layer contains the communication functions specified in this paper which provide additional communication support required for multimedia cooperative application development.
- **Transport layer**: Provides basic transport communication services. The sup-

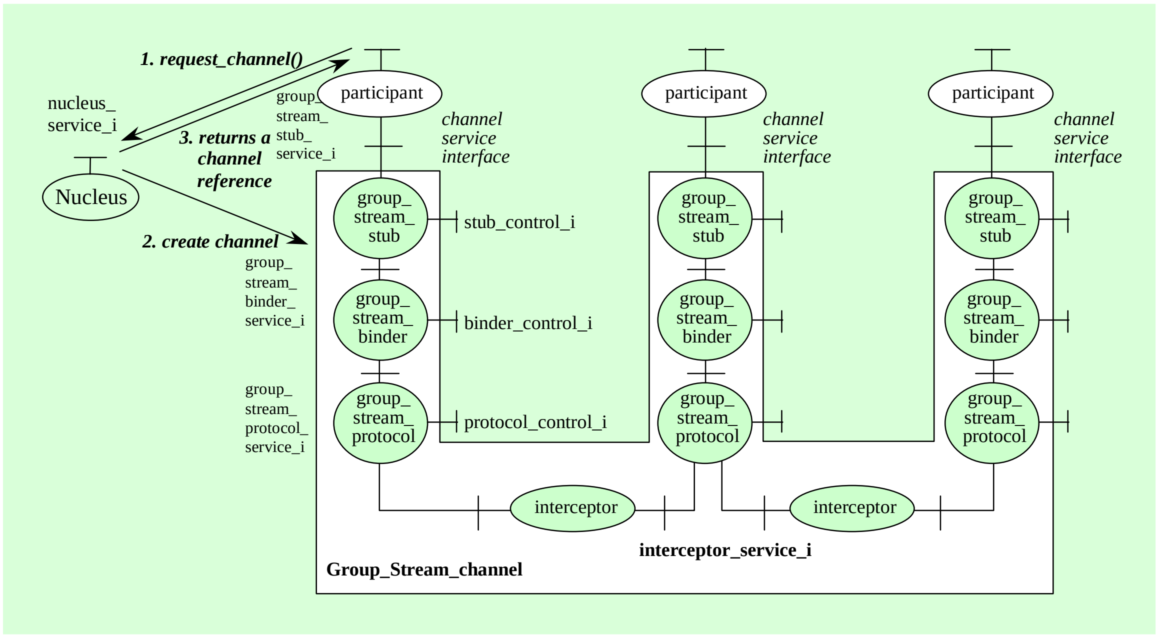

Fig. 4. Configuration of the Group_Stream_Channel object.

Table 5. Specification of the Group_Stream_Binder_control_interface

interface template Group_Stream_Binder_control_interface {
operations
void Modify_Channel_Configuration (inout Bind_Information id, in Bind_Description new_id, in QoS_Information qos, out string result);
void Destroy_Channel (in Bind_References id);
void Monitoring (in Bind_References id);
void Notify_Error (in Bind_References id, out string reason);
void Recovery_Channel (in Bind_References id, out string reason);
behaviour
};

port functions described in the *Support Platform* layer will use these communication services to perform their functionality.

V. CONCLUSION

In the field of distributed processing, thanks to the standardization effort carried

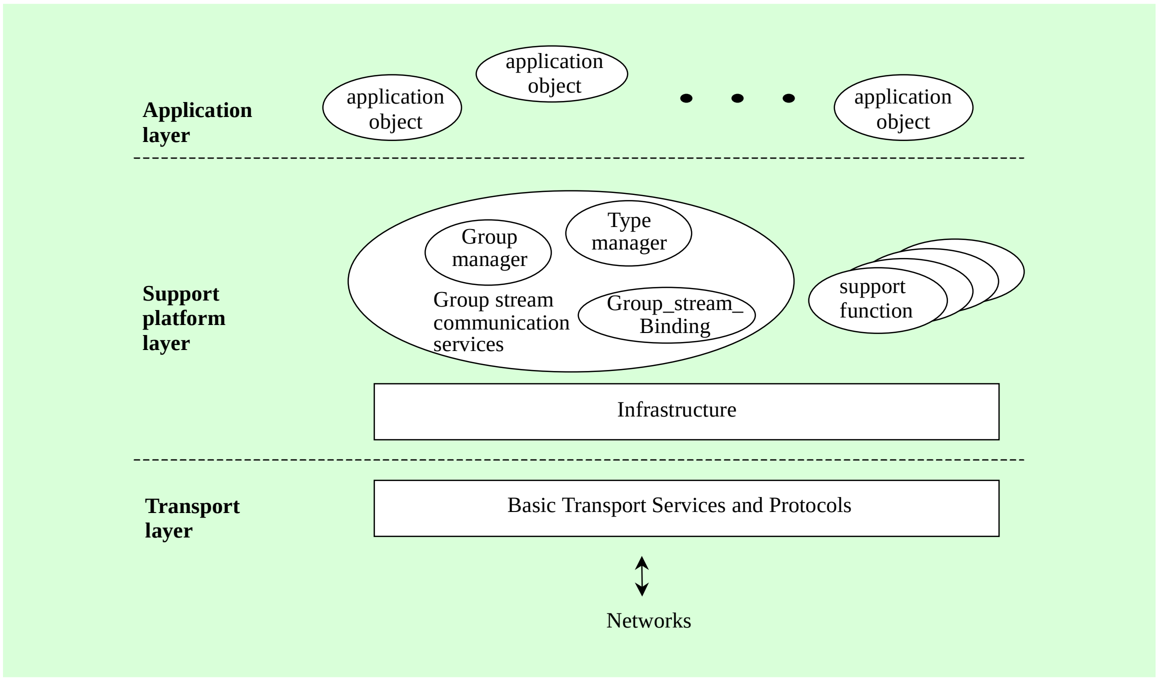

Fig. 5. Development environment of a node.

Table 4. Specification of the Group_Stream_Channel object.

```
channel template Group_Stream_Channel {
supported objects template
    Group_Stream_Stub;
    Group_Stream_Binder;
    Group_Stream_Protocol;
    Interceptor;
initial object template
    Group_Stream_Stub;
behaviour
};
```

out by the organizations, ISO and ITU-T, and numerous research projects carried out by several universities and industrial organizations, the Reference Model of ODP and various *de facto* standards such as ANSA, TINA, CORBA are available [19]. Each one of these standards defines a common support architecture with functions that accommodate difficulties inherent in the building of systems and applications in distributed and heterogeneous environments. In this context, one of the important objective of these architectures is to understand exactly what are the different application requirements and provide suitable facilities needed for the application development. In this way, the support architecture helps to reduce the designer's duplicated effort and the inherent complexity in the development of distributed applications.

After that we analyzed the characteristics of multimedia cooperative applications, we identified, as a result of this analysis, the communication requirements: *Support for Group Interaction, Group Members Management, Group Communication, QoS Guarantees* and *Multimedia data transmission*. From the evaluation of the architectures such as the RM-ODP, ANSA, CORBA, TINA to examine whether they meet these requirements, we identified a lack of functionality: *Dynamic Group Management, Compatibility Checking for Stream Interfaces* and *QoS level in Group Stream Interaction, QoS Management* and *Stream Interface Type Verification*. We then specified a set of communication functions which must be included in an architecture to support this functionality, using the different viewpoints defined in the RM-ODP. However, the communication functions specified in this paper could be extended. Some investigation issues could be considered such as consideration of flexible composition rules in the establishment of binding links between different types of interface, extension of the type verification function for stream interfaces, analysis of requirements of new classes of applications including requirements of real time applications, etc.

We have considered those applications which demand the characteristics of multimedia cooperative interaction between group members as mentioned above. Application designers may use the specified communication services for building this kind of applications. Also in our investigation we included, but not presented in this paper, a study for a particular application: Real-Time Tele-Teaching which is one of the CO-LEARN applications, presenting how the communication services could be used in the design of a specific application.

REFERENCES

[1] ITU-T Rec. X.901 | ISO/IEC 10746-1, *ODP Reference Model Part1: Overview*, DIS, May 1995.

[2] ITU-T Rec. X.903 | ISO/IEC 10746-3, *ODP Reference Model Part3: Architecture*, IS, February 1995.

[3] Tom Rodden and Gordon Blair, "Distributed Systems Support for Computer Supported Cooperative Work," *Computer Communications*, Vol. 15, No. 8, Oct. 1992.

[4] V. Gay, P. Leydekkers, R. Huis in't Veld, "Specification of Multiparty Audio and Video Interaction Based on the Reference Model of Open Distributed Processing," *Computer Networks and ISDN Systems*, Vol. 27, No.8, July 1995.

[5] L. Navarro, W. Prinz, and T. Rodden, "CSCW Requires Open Systems," *Computer Communications*, Vol. 16, No. 5, May 1993.

[6] E. Pastor, D. Fernandez, L. Bellido, "Cooperative Learning over Broadband Networks," *Proceedings of the 6th Joint European Networking Conference (JENC6)*, J. Barbera and J. Kiers, eds., Tel Aviv, Israel, May 1995.

[7] G. Blair and T. Rodden, *The Challenges of CSCW for Open Distributed Processing*, Internal Report MPG-93-24, Lancaster University, 1993.

[8] K. Farooqui, L. Logrippo, and J. de Meer, "The ISO Reference Model for Open Distributed Processing: An Introduction," *Computer Networks and ISDN Systems*, Vol. 27, No. 8, July 1995.

[9] Neil Williams and Gordon S. Blair, "Distributed Multimedia Applications: A Review," *Computer Communications*, Vol. 17, No. 2, February 1994.

[10] APM Ltd., *ANSA: An Engineer's Introduction to the Architecture*, Release TR.03.02., November 1989.

[11] APM Ltd., *A Model for Interface Groups*, Document APM.1001.01, May 1994.

[12] G. Coulson, J. Smalley, and G. Blair, *The Design and Implementation of a Group Invocation Facility in ANSA*, Internal Report MPG-92-34, Lancaster University, 1992.

[13] A.J. Herbert. "An ANSA Overview," *IEEE Network*, January/February 1994.

[14] H.B. Antillanca and D.A. Fuller, "Refining Space-Time Taxonomies of Collaborative Systems," *Proceedings of the Second CYTED-RITOS International Workshop on Groupware (CRIWG96)*, Puerto Varas, Chile, September 1996.

[15] Jong-Hwa Yi and Encarna Pastor, "Communication Support for Cooperative Applications in Open Distributed Processing Systems," *Proceedings of the Second CYTED-RITOS International Workshop on Groupware*, Puerto Varas, Chile, September 1996.

[16] Object Management Group, *The Common Object Request Broker: Architecture and Specification*. OMG Document No. 91.12.1, Revision 1.1, Draft 10, December 1991.

[17] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Revision 2.0, July 1995.

[18] Object Management Group, *Common Facilities Architecture*. Revision 4.0, OMG Document No. 95-1-2, January 1991.

[19] B. Kitson, "CORBA and TINA: The Architecture Relationships," *Proceedings of the TINA95: Intergrating Telecommunications and Distributed Computing – from Concept to Reality*, Vol. 1, Melbourne, Australia, February 1995.

[20] A. Beitz, P. King, and K. Raymond, "Comparing Two Distributed Environments: DCE and ANSAware," *DCE – The OSF Distributed Environment: Client/Server Model and Beyond*, A. Schill, ed., Lecture Notes in Computer Science. No. 731. Springer-Verlag, 1993.

[21] Sally Shlaer, Stephen J. Mellor, *Object Lifecycles: Modeling the World in States*, Yourdon Press Computing Series, Printice-Hall, Inc., 1992.

[22] TINA-C, *Overall Concepts and Principles of TINA*, Version 1.0, February 1995.

[23] TINA-C, *Computational Modeling Concepts*, Version 2.0, February 1995.

[24] TINA-C, *Engineering Modeling Concepts: DPE Kernel Specification*, Version 1.0, December 1995.

[25] TINA-C, *Service Architecture*, Version 2.0, Mayo 1995.

[26] TINA-C, *TINA Object Definition Language (TINA-ODL) Manual*, Version 1.3, June 1995.

[27] Steve Vinoski, "CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments," *IEEE Communications Magazine*, February 1997.

[28] "OSF's Distributed Computing Environment and the requirements," *The Open Systems Newsletter*. Vol. 5, issue 9. September 1992.

[29] A. Vogel, G. Bochmann, R. Dssouli, J. Gecsei, A. Hafid, and B. Kerherve, *On QoS Negotiation in Distributed Multimedia Applications*, Montreal University, Canada, 1993, ftp://ftp.iro.umontreal.ca/publications/1993/IRO-891.ps.

[30] N. Simoni and S. Znaty, "QoS: From Definition to Management." *Proceedings of the IFIP High Performance Networking 92*, A. Danthine and O. Spaniol, eds., 1992.

[31] N.A. Davies, and J.R. Nicol, "Technological Perspectives on Multimedia Computing," *Computer Communications*. Vol. 14, No. 5, June 1991.

[32] A. Karmouth. "Multimedia Distributed Cooperative System, *Computer Communications*. Vol. 16, No. 9, September 1993.

[33] D. Shepherd, D. Hutchison, F. Garcia, and G. Coulson, "Protocol Support for Distributed Multimedia Applications," *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, IBM ENC, Heidelberg, Germany, November 1991.

[34] G. Coulson, G. Blair, N. Davies, and N. Williams, "Extensions to ANSA for Multimedia Computing," *Computer Networks and ISDN Systems*, Vol. 25, No. 3, September 1992.

[35] *Distributed Interactive Multimedia Architecture*, http://www.ansa.co.uk/ANSA/DIMMA/index.html, (October 1995).

[36] TRIBUNE Consortium, *DELTA in PERSPECTIVE-Exploitation of DELTA Projects Results*, TRIBUNE Collection 05, September 1994.

Jong-Hwa Yi received the M.S. degree in Electronics Engineering from Hanyang University, Seoul, Korea in 1990 and the Ph.D degree in Telematics Engineering from Universidad Politécnica de Madrid (Technical University of Madrid), Spain in 1996. She joined ETRI in 1990 and is a senior member of research staff in Protocol Engineering Center (PEC). Her current interest research fields include distributed application architectures, open distributed processing, CSCW applications and object-oriented modeling.

Encarna Pastor is Associate Professor at the Department of Telematic Systems Engineering, Universidad Politécnica de Madrid (Technical University of Madrid), Spain. She received the M.S. and Ph.D. degrees on computer science from the same University in 1977 and 1988, respectively. For the current several years she has been involved in R&D tasks related to architectures and protocols for open system interconnection, distributed applications and computer-supported cooperative work. She is also collaborating with the National Council for Research and Technical Development as scientific coordinator of European Community Programmes (Telematic Applications). Her current interest research fields include communication architectures, open distributed processing and advanced collaborative applications on high performance networks.