Abstract – Recently, the distributed computing environments provide various open multimedia services through telecommunication information networking structure developing based on object-oriented concepts and distributed technology which can be applied new services with a few changes over the existing networks. The existing object model in distributed computing environment has the limitations of the individual object modeling and the complexity of object's management according to large capacity of distributed applications. Our research goal is to design and implement the object group model that can be functionally and efficiently managed the individual objects in a given environment for decreasing the complexity in the distributed software’s management and development. This paper presents the analysis of requirements and functions required to propose the object group model, and depicts the functional structure in details using its analysis. We design the interaction procedures among the components of the object group for management and service functions for our object group model. And we show procedures using interaction procedure diagrams and ETD(Event Tracing Diagram)s. In the implementation of our suggested model, we used the OGTG(Object Group Trader Gateway) developed at our previous researches as a module added to a trader for facilitating the interconnections between/among objects in distributed object groups. Finally, we implemented the object group model combined with the OGTG using Orbix 2.2 which is a CORBA(Common Object Request Broker Architecture)-compliance and OrbixTrader followed specification for trading services of OMG/CORBA, and showed executing procedures of our modules.

Index Terms— TINA, Object Group, Trader, Distributed Object, OGTG(Object Group Trader Gateway)

1 INTRODUCTION

Nowadays, many researchers have been studying the Open Telecommunication Information Networking structure which is structured the telecommunication networking into the hierarchical software model. This model is based on distributed processing technologies, object-oriented modeling[7], and implementing techniques to establish the next generation telecommunication networking architecture in distributed computing environments. These researches have been accomplishing in the Telecommunications Information Networking Architecture-Consortium(TINA-C)[1]. The distributed system is complicated in processing procedures of large capacity of distributed multimedia applications, because it is composed of the sets of the distributed object instances with the complex binding between objects. Hence, the Open Telecommunication Networking has defined the concepts of object group[10], as structuring mechanism for managing the set of computational objects, after this we will call object shortly, and for decreasing the complexity of distributed software’s management.

The existing object model in distributed environment has the limitations of the individual object modeling and the complexity of object’s management[3]. Our research goal is to design and implement the object group model that can be functionally and efficiently managed the individual objects in the distributed computing systems for decreasing the complexity in the distributed software management and development. In this paper, first of all we present the analysis of requirements and functions for making a specification to propose our object group model, and then define the structure of an object group in details using its specifications. The structure of our object group model consists of an Object Group Manager object, two kinds of interface objects for supporting management and services of objects, one or more subgroups, and some repositories. The subgroup has the same components as an object group has.

Also, we consider an interconnection model using the trader[4] for helping easily interactions between/among objects or object group objects(called object group after this) in the distributed environment. But the existing interconnections between/among procedures or modules[8] cause the complex management and service aspects when the procedures for interconnecting objects apply in our proposed model with the hierarchical object group structure. For this
reason, this paper added the OGTG (Object Group Trader Gateway) module between a trader and object groups for solving the problems causing the existing interconnection procedures. Here, the OGTG is one we developed and reported in our previous research[9].

This paper presents the analysis of requirements and functions required to propose the object group model, and depicts the functional structure in details using its analysis.

We design the interaction procedures among the components of the object group for management and service functions for our object group model. And we show procedures using interaction procedure diagrams and ETD (Event Tracing Diagram)s. And we implement using IONA Orbix 2.2 distributed system platform[11] and OrbixTrader which is implemented the trading service of OMG/CORBA[5].

Our paper is organized as follows: Section 2 describes the object group model, the interfaces among the components consisting in an object group, the functional specifications, and the ODL design[6] of an Object Group Manager based on the functional specifications. And we present the interactions between objects in a distributed object group. In section 3, we explain the structure and functions of the OGTG module combining with a trader for supporting efficient interconnections among distributed object groups, and depict the interconnection procedures and the ETD according to the management interconnection between objects included in distributed object groups using the OGTG. Section 4 shows the environments of the object group model we implemented and the monitoring procedure binding distributed object groups via the OGTG and a trader. Finally, we discuss our conclusions and future works.

2 OBJECT GROUP MODELING

A service, an application, or a system in the distributed computing environment consists of lots of objects which is designed based upon telecommunication network to provide a wide range of services over the network. As we mentioned before, the distributed objects cause the complexity for management that TINA has defined the general concepts of an object group. For solving this problem, we design and develop the object group model based on the concepts of TINA’s object group. In this section we describe the object group’s structure, the components containing in an object group, their interactions, and the functionality of the components. And we show the design procedures of an Object Group Manager using TINA-ODL (Object Definition Language)[6].

2.1 Object Group Model

We propose the structure of an object group based on the definition of the object group described in TINA[1]. An object group defines a distributed unit, as a set of objects, for executing service or management functions. And each object group is controlled by an Object Group Manager object, called an OG Manager after this, and an OG Manager manages the other objects except oneself object in the object group, and this object is the first served object can be instantiated in an object group. Also, we defined an object group as a hierarchical structure, which can be included one or more subgroups in itself. The main goal of object grouping through abstraction of objects is not only to decrease the complexity of distributed software’s development and management, but also improve an efficiency of object management in the object-oriented model.

The structure of object group consists of the following components; when an object in an object group requires to desiring objects in the others, these objects have to bind each other via the Management Interface object, called M-I/F, for object management, and the Service Interface object, called S-I/F, for application service, respectively. An OG Manager is responsible for managing all objects in an object group. An Object Instance Repository object, called an Object Instance Repository, stores and maintains the information of objects contained in the same object group or subgroups. A Security Repository Object, called Security Repository, has the information of access rights of target objects. The information for mapping between CORBA (Common Object Request Broker Architecture) objects and computational objects take into an Implementation Map object, called Implementation Map. A subgroup deals with as a nested object structure, The subgroup can be had the same that components which an object group has. Figure 1 illustrates components in the object group model and the interactions among them.

![Figure 1. The Structure of the Object Group Model](image)

The interactions among components in an object group explain as follows;

1. M-I/F invokes Object Instance Repository to search the information about objects in an object group, and after Object Instance Repository searches the relating information and then returns its searching result to M-I/F.

2. M-I/F invokes Security Repository to check whether the object can access a specified object in the object group, and as a result, Security Repository returns the result of access possibility to M-I/F after checking access right of the desiring object.

3. M-I/F invokes OG Manager to process the following requirements like creation, deletion, activation, and deactivation of objects or subgroups. Creation and modification information about objects is saved into Object Instance Repository, and modification and searching of access condition are saved into Security Repository, respectively. And
then OG Manager returns the processing result of the requirements to M-I/F.

④ OG Manager invokes Object Instance Repository to create/delete of objects, change access permissions, or search information about the target object. And then Object Instance Repository returns the processing result about above requirements to OG Manager again.

⑤ OG Manager invokes Security Repository to create, delete, and update the security conditions about objects, and to search the security condition about the target object. Also, OG Manager invokes Security Repository to search the ACL (Access Control List) object and is returned the proper information or the results about requesting by Security Repository.

⑥ When OG Manager creates objects or subgroups in an object group, OG Manager requests the mapping information of object from Implementation Map, and then Implementation Map returns the suitable mapping information to OG Manager.

⑦ OG Manager can directly manage the creation, deletion, activation, and deactivation of objects requested.

⑧ The upper OG Manager notifies that the external object requests the management service to M-I/F in a nested subgroup. Then M-I/F in subgroup returns the results processed in subgroup to the upper OG manager.

Figure 2 shows the functional class diagram of the object group’s components using OMT[7].

### 2.2 The Design of Object Group Model

This section describes the design procedures of the object group model based on earlier mentioned functionality analysis and structure of the object group model as we defined before. An OG manager playing an important central role in model, designed using TINA-ODL[6] is following;

```c
typedef sequence <Impl_Info> Impl_Info_List;
/* attribute name and value of computational object */
struct Attr {
  String AttrName;
  Any attrValue;
};
typedef sequence <Attr> Attr_List;
/* the structure for managing Object Instance Repository */
struct Object_Info {
  Template_Name io_template_name;
  Object_Reference obj_ref;
  char object_status;
  char read_permission_flag;
};
typedef sequence <Obj_Info> Obj_Info_List;
```

```c
typedef sequence <Attr> Attr_List;
```

```c
typedef sequence <Obj_Info> Obj_Info_List;
```
struct Security_Tag {
    Req_Entity_Id entity_id;
    /* ID of requesting entity*/
    Operation_Name_List allowed_operation_list;
    /* a kind of allowed operations */
    typedef sequence<Security_Tag> Security_Tag_List;
    attribute Object_Reference obj_ref;
    /* life-cycle service of computational object*/
    Object_Reference create(in Template_Name co_template_name,
                            in Req_Entity_Id entity_id);
    void destroy(in Object_Reference obj_ref,
                 in Req_Entity_Id entity_id);
    void enable(in Object_Reference obj_ref);
    void disable(in Object_Reference obj_ref);
    /* life-cycle service of subgroup */
    Object_Reference create_subgrp(in Template_Name mo_template_name,
                                    in Req_Entity_Id entity_id);
    void destroy(in Object_Reference obj_ref,
                 in Req_Entity_Id entity_id);
    void enable(in Object_Reference obj_ref);
    void disable(in Object_Reference obj_ref);
    /* management service of Object Instance Repository */
    void regist(in DN distinguished_name);
    void withdraw(in DN distinguished_name);
    /* Security service */
    void add_access_rule(in DN distinguished_name);
    Security_Tag lookup_access_rule(in DN distinguished_name);
    void remove_access_rule(in DN distinguished_name);
    void update_access_rule(in DN distinguished_name);
    Security_Tag_List m_lookup_ACL(in DN distinguished_name);
};

As we shown above, an OG manager has functions that could be called as methods by internal or external objects. Also, the instance of OG manager can invoke the interface functions of the following objects for management, security, and creation of object or subgroup. An OG manager invokes create_obj_info(), delete_obj_info(), update_obj_info(), and lookup_obj_info() in Obj Repos infinity of Object Repository interface for object management, and create_ACL(), delete_ACL(), and lookup_ACL() in Security Repos infinity of Security Repository interface for object security, respectively. And finally this manager invokes create_impl_info(), update_impl_info(), delete_impl_info(), and lookup_impl_info() in an Implementation Map interface for object creation.

3 THE INTERCONNECTION DESIGN OF OBJECTS

In this chapter, we describe interconnection procedures for supporting interoperability between/among local and remote objects or object groups. First of all, we have to consider security problems for checking access right of the upper object group before binding between objects in object group. To solve this problem, we had developed an OGTG(Object Group Trader Gateway) as mediator between a trader and object groups, and reported by [9]. So, in this chapter, we will simply mention about the structure and interactions of 3 basic functions of OGTG[8,9], and illustrate the management procedures of the OGTG by using Event Trace Diagram.

3.1 The OGTG(Object Group Trader Gateway)

The interconnection procedures of objects adopted on object group model can complicate rather than one under the existing object model[9] because our object group model has the nested object group structure. That is, we are not easy to directly be applied to interconnection procedures for the management of objects or subgroups in object groups due to following two problems. One is that an arbitrarily object can't be directly reached to subgroups or objects nested in object groups by using the existing trader. And the other can not check the access rights about objects or object groups, while arbitrary objects or subgroups could be accessed or managed by others. Hence, We use an OGTG, as an extended trading model, developed to solve these problems.

This OGTG consists of an OGTG manager and a Mapping Repository. We show the basic structure of the OGTG in Figure 3. A Mapping Repository is constructed by mapping repository object and the storage of a hierarchical M-I/F mapping information to easily authenticate hierarchical authorizations of objects. An OGTG manager is responsible for interfacing between/among object groups, a Mapping Repository, and a trader. A Mapping Repository executes appropriate operations according to requests of an OGTG manager. The OGTG is functionally divided into an OGTG manager module and a Mapping Repository module.

3.2 The Interaction Procedures With OGTG

In this section, we describe the interaction procedures between an OGTG and a trader, in details, query, and export or import. And these interactions are specified by invoking functions between an OGTG manager and a Mapping Repository in OGTG. The categorization of these procedures consists of 3 kinds of functions; Export, Query, and Withdraw.
In the first procedure, an OG Manager invokes an OGTG for exporting M-I/F in itself object group. And then, OGTG manager adds the information of M-I/F exported to a Mapping Repository, in the same time, exports its information to an existing trader. If the exported object is not M-I/F object, an OGTG manager skips a Mapping Repository and invoke a trader for exporting. Otherwise, an OGTG manager stores the Offer-Id information to a Mapping Repository. Finally, the OGTG manager returns Offer-Id returned to the upper OG manager. The Figure 4 shows the interaction procedures of Export function.

![Figure 4. The Interactions of Export function](image)

The Figure 5 describes the Withdraw procedures of M-I/F requested from OG Manager. When OG Manager requests Withdraw operation of the M-I/F to the OGTG, OGTG manager removes the mapping information of M-I/F from Mapping Repository, in the same time, sends withdraw message to a trader. Finally, a trader reports a return-value to Exporter by way of the OGTG.

![Figure 5. The Interactions of Withdraw Function](image)

The Figure 6 explains the interaction procedures of query that can obtain the reference of an appropriate M-I/F which is requested from an OGTG manager. In query procedure, unlike Figure 4 and 5, the OGTG firstly checks if M-I/F matching with Offer-Id provided from a trader is one belonging to a subgroup. If M-I/F is not one in a subgroup, the Offer-Id is directly passed the client(requesting object). Otherwise, an OGTG manager obtains the M-I/F reference of upper-level object group, and checks that is possible to access the upper-level object group by a client object, and returns offer-Id to the client if possible to access.

![Figure 6. The Interactions of Query function](image)

3.3 The Interconnections for Object Management and Service

The interconnection procedures for managements and services between objects or object groups in distributed systems are based on the commercial OrbixTrader environments applying for OMG Trading Object Service Specification. Our extended trading servicing procedures are specified into the interconnections for management and service. The management interconnection procedures are occurred when objects or subgroups in object groups are created, destructed, activated, or inactivated by others. The service interconnection procedures are occurred when objects are connected with one another for servicing distributed application. However, we discuss only the management interconnection procedures because our suggesting an OGTG module has the main goal for management, that is, checking the access right for management of objects in object groups.

For showing management of objects, we use the ETD(Event Tracing Diagram) for specifying interconnection procedures.

In this section, we will discuss only the procedure of the object creation for saving our paper spaces, because the others (destruction, activation, inactivation of objects) are very similar to this procedure. In details, let us consider when the object CO-1 in ObjectGroup-1(OG-1) is about to create the object CO-21 in SubObjectGroup-2(SubOG-2) of the other ObjectGroup-2(OG-2), as shown in Figure 7. Here, we can illustrate two ways of the interconnection procedure of objects appearing in one object group and in the other object groups. The number shown in Figure 7 depicts a sequence of the management procedures, while the object (CO-21) is creating.

![Figure 7. The Procedure for Object creation(CO-21)](image)
possible to access an upper-level object group, OG-2, before accessing CO-21 in SubOG-2 using returned reference. In details, the reference of the M-I/F in an upper-level object group is obtained via the Mapping Repository in OGTG. And then, if possible to access the target object, OGTG returns the reference of M-I/F in SubOG-2 to the CO-1 in OG-1. The CO-1 of OG-1 requests an object CO-21 creation to the M-I/F in SubOG-2 using M-I/F reference returned. The M-I/F in SubOG-2 confirms from Security Repository whether the CO-1 can access the objects in SubOG-2. If it is possible to access SubOG-2, the M-I/F in SubOG-2 takes a message of CO-21 creation over the OG Manager in the same subgroup. After the OG Manager checks whether CO-21 is in Object Instance Repository of SubOG-2, if not, it creates CO-21 referring a template of CO-21 from Implementation Map.

**4. The IMPLEMENTTATION OF OBJECT GROUP MODEL**

We implemented the modules for a distributed object computing environment that could support the object groups and the OGTG using Orbix 2.2 and Orbixtrader 1.0 of IONA Co in accordance with interconnection design suggested in chapter 3. We specified the interfaces of objects in modules by Orbix IDL and compiled the defined interfaces using Orbix IDL compiler. The module source code is written in Visual C++ 4.2 and C++ 4.1 running Solaris. Figure 9 represents our development platform for implementing above modules.

**4.1 The Object Group Module**

The implementing of relating modules in our object group model is as follows; M-I/F, S-I/F, OG Manager, Security Object, Object Instance Repository, ACL object, and Implementation Map. First, the ODL of these components in object group designed by TINA specification are converted into IDL file for implementing of our model on distributed environments. And then the CO-21 created returns its reference to OG Manager in SubOG-2. The OG Manager registers the information of CO-21 to Object Instance Repository, and stores its authorization information to Security Repository. After the OG Manager exported the information relating to CO-21 to a trader via OGTG, it returns the reference of CO-21 returned from a trader to M-I/F in SubOG-2. And continually, M-I/F informs the information returned to the CO-1. Here, the sequence numbers from 8 to 17 illustrate the interconnection procedures between objects in an object group.

Figure 8 is shown as an ETD(Event Tracing Diagram) presented by a sequence of operations and the related objects for creating object CO-21. When an client object requests CO-21 created, this object CO-21 will be used as an object supporting one of distributed services.
The IDL specification is processed by Orbix IDL compiler, which generates a header file for the CORBA object, stub code for linking into the client, and skeleton code for the server object. Here, the server skeleton code is updated by adapting the detailed design that reflects the interconnection procedure and ETD considered in the service face of object. The server objects about all components of an object group are accomplished by updating the skeleton code.

For creation of a new object, firstly the execution procedure of M-I/F and OG manager including in an object group are shown in Figure 10 and 11 respectively. Also, Figure 12 shows the execution procedures of the others except above two objects including in an object group.

4.2 The OGTG Module

The implementation of the OGTG module converts the ODL about the OGTG Manager and a Mapping Repository into IDL file, and the IDL specification is mapped into C++ code using Orbix IDL compiler running Solaris, then the module code is accomplish by adding detailed code. Figure 13 displays the execution procedure of the OGTG Manager interacts with the modules of an OGTG.

5. CONCLUSION

Recently, the distributed computing environments have been requiring the quantitative as well as qualitative requirements for multimedia stream services. The researches of Open Information Networking architecture have been achieved on object-oriented software that can apply new services with a few changes the existing network. The distributed applications should be executed as the unit of object and/or object group categorized components or functionalities. For this reason, we designed and implemented the object group model achieving a goal for decreasing the complexity in management and development of distributed software.

In main contents, we firstly proposed the object group model, defined the functions about the components of an object group, and designed all components by TINA-ODL in accordance with the definition of functionality. We designed the interconnection procedures and ETD according to the management interconnection between objects and object groups using the OGTG module developed at our previous research for solving the security check problem between object groups. We use of Orbix 2.2, Visual C++, and C++ running Solaris for implementing the object group module and the OGTG module. Finally, this paper showed the execution procedures of the components in object groups and the OGTG Manager for interconnecting among them, as final execution results.
In future, we are interested in area of world-wide distributed system based on federation trader in distributed computing environments. These researching contents can be included studies on the extension of functionality about the components of an object group, the verification of functionality, and the development of federation trading services for supporting the hierarchical object group or distributed replicated object model.

ACKNOWLEDGMENTS

This work was supported by research grants from Won-Kwang University in 2000.

REFERENCES


Won-Jung Lee is received his B.S degree from Chen-Buk University, Korea, in 1991, M.S. degree in computer engineering from Won-Kwang University, Korea, in 1998. He is currently a PhD student at dept. of computer and communication engineering in the same university. His interesting areas include distributed object model, multi-database systems, object-oriented model.

Chang-Won Jeong is received B.S. and M.S. degrees in computer engineering from Won-Kwang University, Korea in 1996 and 1998 respectively. He is currently working as a PhD student at dept. of computer and communication engineering, Won-Kwang University in Korea. Her current research interests are in distributed object model, trading object services and wide-area distributed systems.

Myung-Hee Kim is received B.S. and M.S. degrees in computer engineering from Won-Kwang University, Korea, in 1993 and 1996, respectively. Currently, she is working toward the Ph.D degree at dept. of computer and communication engineering, Won-Kwang University in Korea. Her current research interests are in distributed real-time computing, system optimization, operating systems.

Su-Chong Joo received his BS degree in computer engineering from Won-Kwang University, and M.S. and Ph.D. degrees at dept. of computer science & Engineering from Chung-Ang University, Korea, Currently, he is associate professor at dept. of computer and communication engineering in Won-Kwang University. His main research interests include distributed real-time computing, distributed object modeling, system optimization, multimedia database systems. He served as an International Program Committee member for IASTED International Conference on parallel, distributed computing and network (IASTED’97, Singapore). From Jul. 1993 to Aug. 1994, he was a Post-Doctoral Fellow at dept. of electrical and computer engineering in Univ. of Massachusetts, Amherst, USA. He is a member of KISS, IASTED, and IEEE computer society.