International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.2 (2014), pp.235-244
http://dx.doi.org/10.14257/ijmue.2014.9.2.23

Interconnections of Object Management for U-healthcare Services

Haeng-Kon Kim

School of Information Technology, Catholic University of Daegu, Korea
hangkon@cu.ac.kr

Abstract

Enterprise Information Technology (IT) professionals have reached the practical limits of two-tier architectures. Facing growing pressure to deliver broad-reaching application functionality quickly and cost-effectively, IT departments are turning to distributed object computing as the adaptive software architecture on which to build applications that meet these challenging business requirements. Besides, as a growing number of sophisticated services are provided over the Internet, the need arises to coordinate application objects into transactions, in such a way as to enable functionality and data to be shared across applications and over multiple platforms. The distributed computing environments provide various open u-healthcare 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.

In this paper, we 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 for u-health care services object management. We also presents the analysis of requirements and functions required to propose the object group model for u-healthcare, 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 ETDs(Event Tracing Diagram)s.

Keyword: *Distributed Computing Environments, u-healthcare multimedia services, distributed software's management, ETD (Event Tracing Diagram)*

1. Introduction

The systems-engineering and data-processing industries have experienced two major revolutions in the recent era. The first revolution was caused by the increasing complexity of software - resulting in the introduction of new, structured software engineering practices. Object-oriented computing is a product of this revolution. The second major revolution is the advent of distributed computing. Although the mainframe is still central in many corporations, it is steadily being the replaced by legions of connected personal computers. It is apparent that these two computing paradigms will be the most powerful forces affecting the future of the software industry. Corporations have realized that the monolithic and expensive mainframe-type computers will not deliver the performance demanded by tomorrow's diverse information systems The existing object model in distributed environment has the limitations of the individual object modeling and the complexity of object's management [1]. 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

ISSN: 1975-0080 IJMUE
Copyright ⓒ 2014 SERSC

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 [2] 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[3] cause the complex management and service aspects when the procedures for interconnecting objects apply in our proposed model with the hierarchical object group structure. In this paper, we also presents the analysis of requirements and functions required to propose the object group model for u-healthcare, 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 ETDs(Event Tracing Diagram)s.

2. Object Group Model for U-healthcare

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.

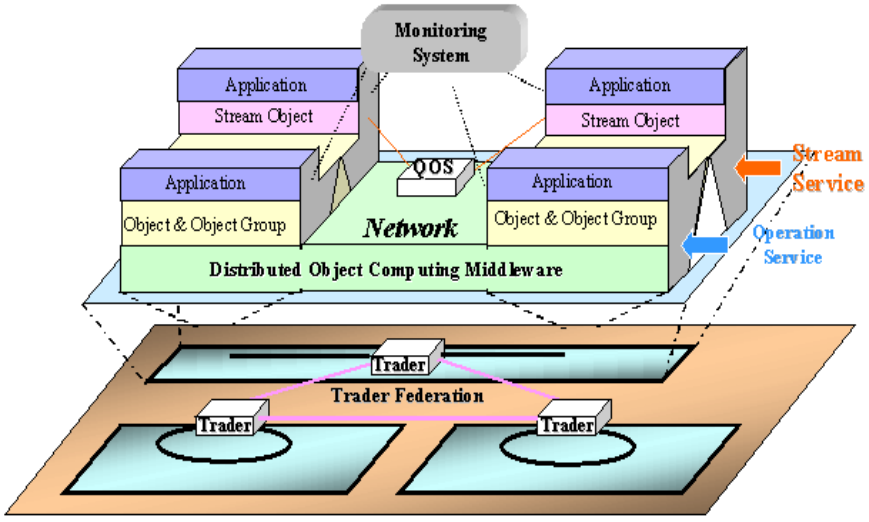

Figure 1. The Structure of the Object Group Model

Copyright ⓒ 2014 SERSC

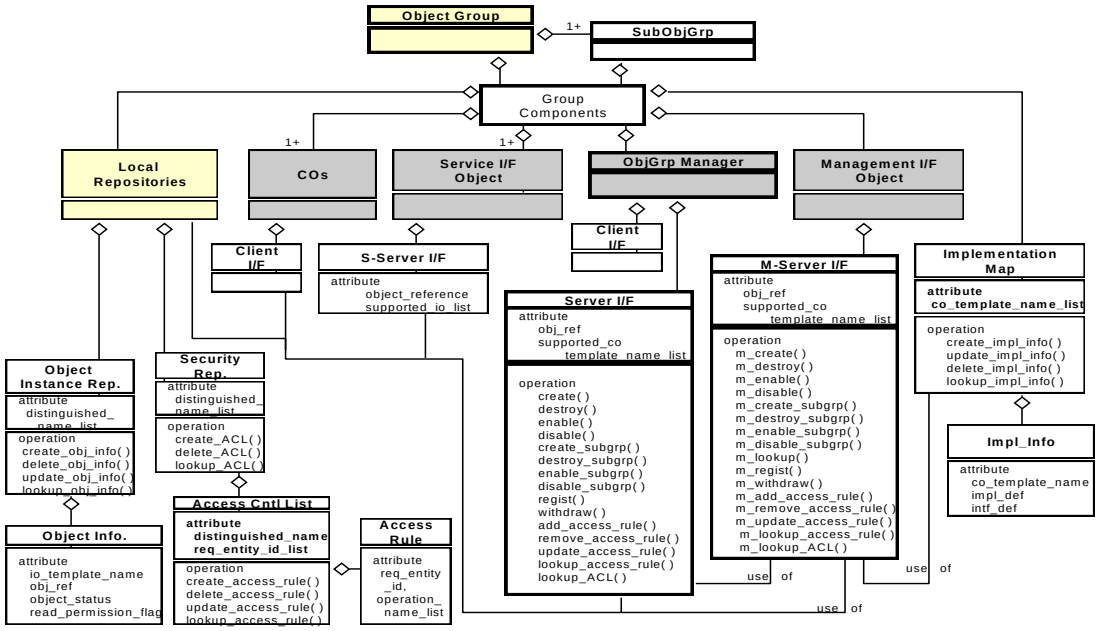

Figure 2. Functional class diagram of the object group's components using UML

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**.**

3. The Interconnection Design of U-healthcare Object

3.1 Basic Concept

The interconnection procedures of objects adopted on the object group model can complicate rather than one under the existing object model [3], 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 not 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 target objects or object groups, while arbitrary objects or subgroups could be accessed by others. Hence, we use the OGTG(Object Group Trader Gateway 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.

Copyright ⓒ 2014 SERSC

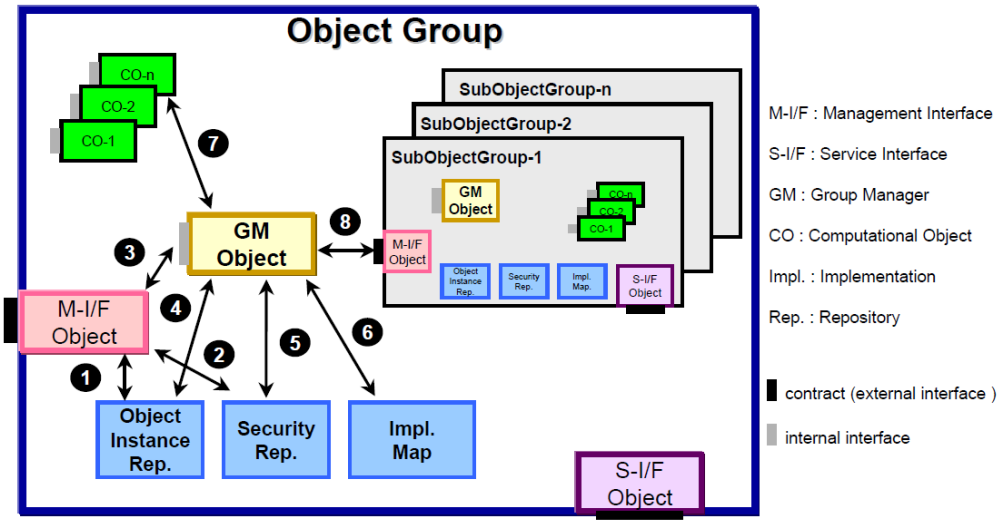

Figure 2. Basic structure of the OGTG

3.2 The Interaction Procedures With OGTG

The interactions are specified by invoking functions between an OGTG manager and a Mapping Repository in an 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, an 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 invokes a trader for exporting. Otherwise, an OGTG manager adds the mapping information relating with M-I/F of the upper-level object group to a Mapping Repository concurrently. The invoked trader returns *Offer-Id* of the exported object, and OGTG manager stores *Offer-Id* information to a Mapping Repository. Finally, an OGTG manager returns Offer-Id returned to the upper-level OG manger. The Figure 3 shows the interaction procedures of Export function.

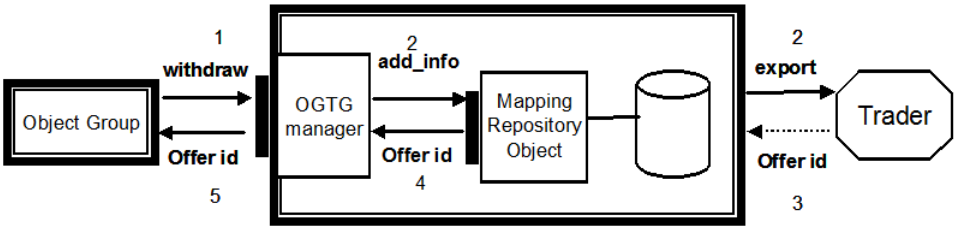

Figure 3. The Interactions of Export Function

The Figure 4 describes the Withdraw procedures of M-I/F requested from an OG Manager. When the 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 a withdraw message to a trader. Finally, a trader reports a return-value to Exporter by way of the OGTG.

Copyright ⓒ 2014 SERSC

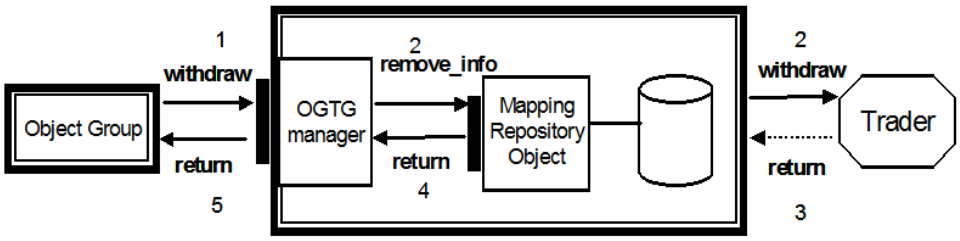

Figure 4. The Interactions of Withdraw Function

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 will discuss only the management interconnection procedures because our suggesting 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 5. Here, we can illustrate two ways of the interconnection procedure of objects appearing in one object group and in the other object groups. The numbers shown in Figure 5 depict a sequence of the management procedures, while the object (CO-21) is creating.

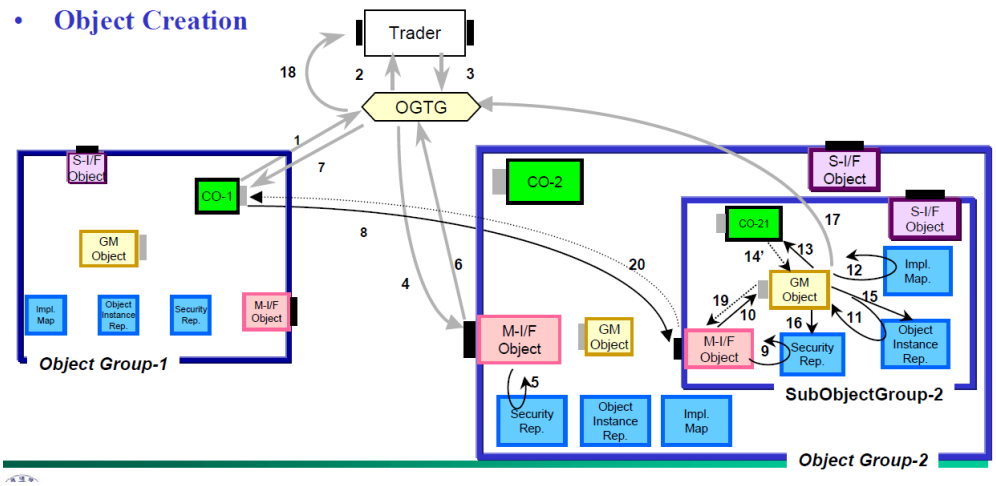

Figure 5. The Procedure for Object Creation

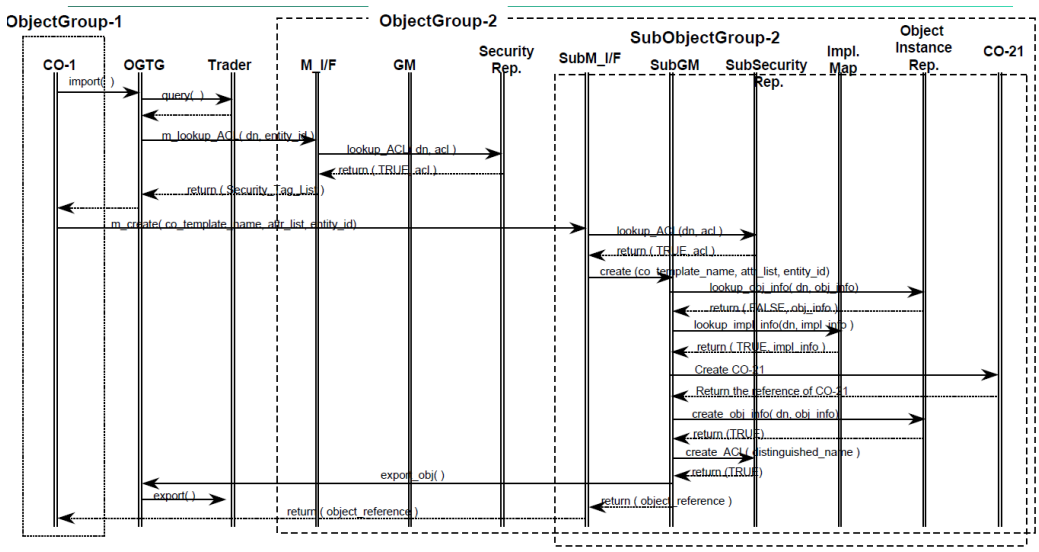

Figure 6. The ETD for Creating An Object

4. The Implementation of U-healthcare 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 Oribxtrader 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 then the CO-21 created returns its reference to OG Manger 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 7 is shown as an ETD 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.

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.

 Copyright ⓒ 2014 SERSC

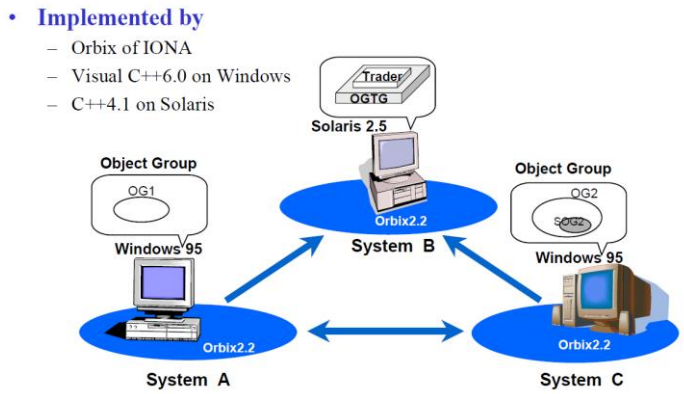

Figure 7. The Development Platform

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 a OG manager including in an object group are shown in Figure 8 and 9 respectively. Also, Figure 10 shows the execution procedures of the others except above two objects including in an object group.

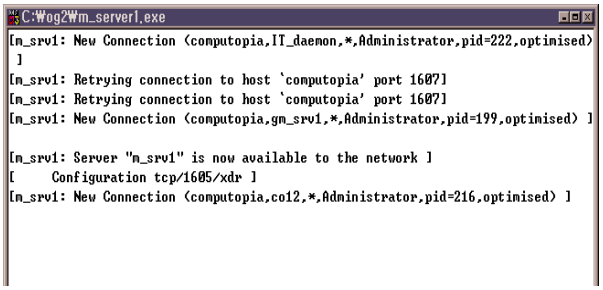

Figure 8. The Execution Procedure of M-I/F

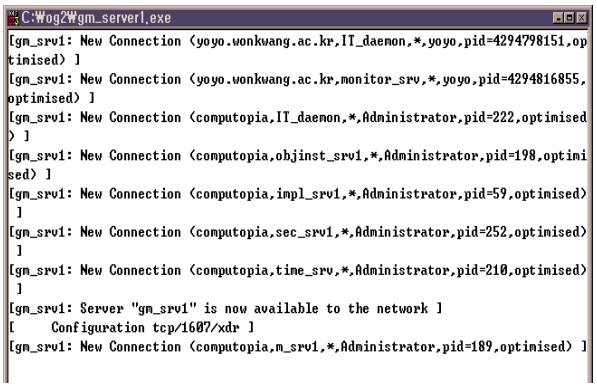

Figure 9. The Execution Procedure of an OG Manager

Copyright ⓒ 2014 SERSC

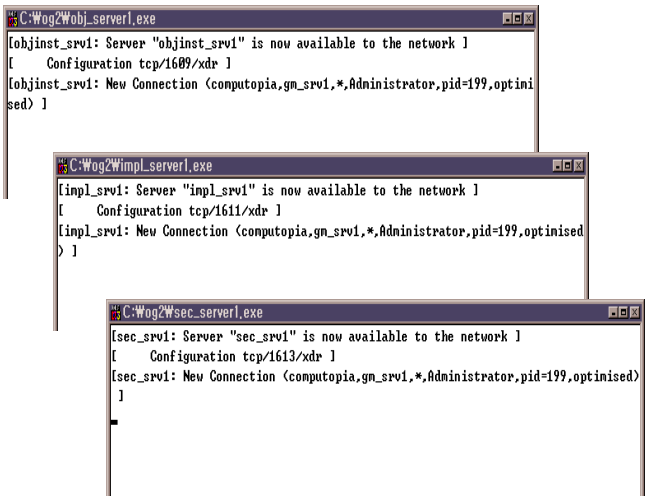

Figure 10. The Execution Procedures of the others objects in an Object Group

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.

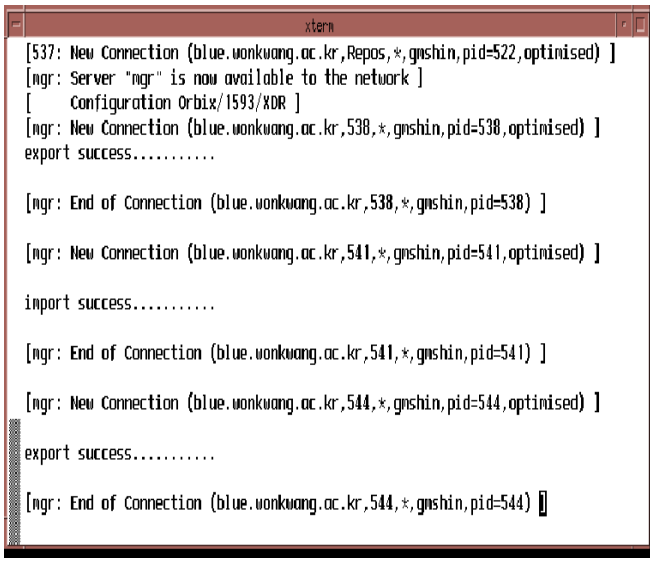

Figure 11. The Execution Procedure of the OGTG Manager

 Copyright ⓒ 2014 SERSC

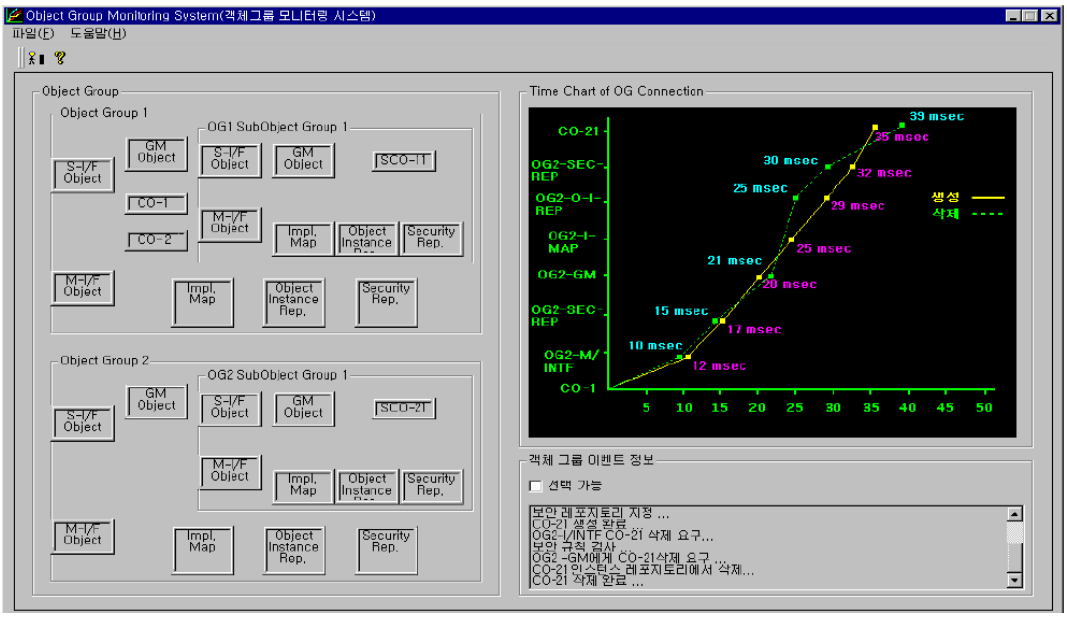

Figure 12. The Execution of Interconnections of Object Management for U-healthcare Services

5. Conclusion

The client application of u-healthcare interacts with the client-side ORB(Object Request Broker) directly. The client application in this context is the combination of the custom code and the stub / skeleton code generated by the **IDL(**Interactive Data Language) compiler. Stub / skeleton code uses the ORB interface to communicate with the ORB directly.

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.

Copyright ⓒ 2014 SERSC

Acknowledgements

"This research was Supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the CITRC (Convergence Information Technology Research Center) support program (NIPA-2014-H0401-14-1008) supervised by the NIPA (National IT Industry Promotion Agency)."

This research was also supported by the International Research & Development Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (Grant number: K 2012057499)."

References

[1] M. Rodriguez, J. Favela, V. Gonzalez and M. Munoz, "Agent Based Mobile Collaboration and Information Access in a Healthcare Environment", Proceedings of Workshop of E-Health: Applications of Computing Science in Medicine and Health Care, ISBN: 970-36-0118-9, Cuernavaca, Mexico, **(2003)** December.
[2] J. Riekki, O. Davidyuk, J. Forstadius, J. Sun and J. Sauvola, "Enabling Context-Aware Services for Mobile Users", MCCSIS 2005, **(2005)** April.
[3] Y. Huang and H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment", Wireless Networks, **(2004)**.
[4] T. L. Pham, G. Schneider, S. Goose and A. Pizano, "Composite Device Computing Environment: A Framework for Situated Interaction using Small Screen Devices", Personal and Ubiquitous Computing, **(2001)**.
[5] L. Kagal, V. Korolev, H. Chen, A. Joshi and T. Finin, "Centaurus: A framework for intelligent services in a mobile environment", Distributed Computing Systems Workshop, International Conference, **(2001)** April, pp. 195-201.
[6] C. -S. Shin, C. -W. Jeong and S. -C. Joo, "Construction of Distributed Object Group Framework and Its Execution Analysis Using Distributed Application Simulation", Embedded and Ubiquitous Computing: International Conference EUC 2004, **(2004)** August, pp. 724-733, 756.
[7] D. -S. Kim, C. -W. Jeong and S. -C. Joo, "Implementation of Healthcare Application Service in Mobile Collaboration Environment", Proceedings of Korea Computer Congress 2006, vol. 33, no. 1(D), **(2006)** June 21-23, pp. 88-90.
[8] D. Agarwal, C. McParland and M. Perry, "Supporting Collaborative Computing and Interaction", Proceedings of the Grace Hopper Celebration of Women in Computing 2002 Conference, **(2002)** October 9-12, Vancouver, Canada.
[9] A. Heinemann, J. Kangasharju, F. Lyardet and M. Mühlhäuser, "iClouds -Peer-to-Peer Information Sharing in Mobile Environments", International Conference on Parallel and Distributed Computing (Euro-Par 2003). Klagenfurt, Austria, **(2003)** August 26-29.
[10] Sun Microsystems. jxme: JXTA Java Micro Edition Project, http://jxme.jxta.org/.
[11] G. Kortuem, j. Schneider, D. P. Thaddeus, G. C. Thompson, S. Fickas and Z. Segall, "When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-peer Computing in Mobile Ad hoc Networks", In First International Conference on Peer-to-Peer Computing, Linkoping Sweeden, **(2001)** August 27-29.
[12] K. H. Kim, M. Ishida and J. Liu, "An Efficient Middleware Architecture Supporting Time-triggered Message-triggered Objects and an NT-based Implementation", In Proceedings of the IEEE CS 2nd International Symposium on Object-oriented Real-time Distributed Computing(ISORC'99), **(1999)**, pp. 54-63.
[13] F. Baboescu and G. Varghese, "Scalable Packet Classification", IEEE/ACM transactions on networking, vol. 13, no. 1, **(2005)** February.
[14] T. V. Lakshman and D. Stidialis, "High speed policy-based packet forwarding using efficient multi-dimensional range matching", in Proc. ACM SIGCOMM, **(1998)** September.

 Copyright ⓒ 2014 SERSC