マルチメディア通信と分散処理 63−13
グループウェア 5−13
（1994. 1. 28）

ODPビューポイントに基づく分散システムの設計法とその通信システムへの適用

藤長 昌彦　　加藤 聰彦　　鈴木 健二

国際電信電話 (株) 研究所

ISOおよびITU-Tで標準化されているODPにおいては、分散システムの複雑さに対処するために、5つのビューポイントを導入している。本論文では、まず、ODPビューポイントに基づいて分散システムを設計する方法について述べる。本方法では、要求分析において、エンタプライズビューポイントから、機能設計において、情報、コンピュテーショナル、エンジニアリングの各ビューポイントから、詳細設計において、テクノロジビューポイントから、それぞれシステムの仕様を作成することにより、分散システムの設計を進めるものである。さらに本論文では、本設計法を、複数の計算機から構成されるMHSシステムの設計に適用した結果についても述べる。

Design Method of Distributed System based on ODP Viewpoint Approach and its Application to Telecommunication System

Masahiko FUJINAGA　　Toshihiko KATO　　Kenji SUZUKI

KDD R&D Laboratories

Ohara 2-1-15, Kamifukuoka, Saitama 356, JAPAN

In the Open Distributed Processing studied by ISO and ITU-T, five viewpoints are defined to avoid dealing with the full complexity of distributed systems. First, this paper describes a design method of a distributed system based on the ODP viewpoint approach. The method uses the specifications of the target system from the individual viewpoints in the requirement analysis phase, functional specification phase and detailed design phase. This paper also shows the result of the applicaton of the design method to the design of a distributed MHS system supporting the Message Transfer Agent function.

1 Introduction

Along with the improvement of micro processors and high speed networks, a distributed system, which is built by using multiple computers connected through a high speed local area network, becomes useful to realize various types of systems such as a telecommunication system and a scientific computation system. A distributed system has potential advantage to achieve high performance by partitioning the load of the system among computers. High reliability can be also accomplished because it is possible to isolate the failed computers from the system and to continue its operation. However, the design of a distributed system generally tends to be more complicated compared to a conventional centralized system, because the system designer needs to address not only the application specific issues but also the distribution of computers.

In order to aim at the standardization of distributed processing, ISO/IEC JTC1 SC21/WG7 and ITU-T SG7 study the Open Distributed Processing (ODP) and are currently standardizing the Basic Reference Model of ODP (RM-ODP) [1, 2, 3]. To avoid dealing with the full complexity of distributed systems, RM-ODP defines five viewpoints: *enterprise*, *information*, *computational*, *engineering* and *technology viewpoint*. Each viewpoint gives different abstraction of a distributed system and can reduce the complexity of describing the distributed system.

The viewpoint approach of RM-ODP is proposed as a framework for the design of a distributed system [3], and some previous works give the design methods of distributed systems based on the ODP viewpoint approach [4, 5]. In those design methods, the design proceeds by giving the specifications of the target system from the individual viewpoints. In the previous works, however, the detailed design method has not been fully specified or all of the five viewpoints are not taken into account.

In this paper, we describe a detailed design method of a distributed system based on the ODP viewpoint approach. Our method applies the five viewpoints to the design of a distributed system, and shows what specifications for the target system are made from the individual viewpoints and how the specifications are related. It also takes account of *platform* which provides the distribution support [6, 7, 8], and gives the design method for both the platform and the application specific functions on the platform.

In this paper, we also apply our method to the implementation of a telecommunication system. As an example of a telecommunication system, we take an MHS system supporting the Message Transfer Agent (MTA) function which relays messages between other MHS systems according to the X.400 protocols.

2 Design Method of Distributed System

2.1 ODP Viewpoints

The RM-ODP defines the following five viewpoints.

Enterprise Viewpoint
The enterprise environment including objectives, management policies, roles of users and requirements of the system are described from this viewpoint.

Information Viewpoint
This viewpoint focuses on the structure of the information elements which the system handles, and the information flow stating how the information elements are modified or remain unchanged in the system.

Computational Viewpoint
The functional modules of the system are observed from this viewpoint. The interface of each functional module is given as its signature, that is, the name of the procedure and the data types of its parameters.

Engineering Viewpoint
This viewpoint specifies how the distribution transparency is realized.

Technology Viewpoint
This viewpoint specifies how the system will be implemented using hardware and software selected from the available products.

2.2 Design Method based on Viewpoint Approach

The top-down design approach for distributed systems generally takes three phases: requirement analysis phase, functional specification phase and detailed design phase. In order to clarify the design method based on the ODP viewpoint approach, this section describes how viewpoints are applied to those design phases, what needs to be described in the viewpoint specifications, and how the viewpoint specifications are related with each other.

2.2.1 Requirement Analysis Phase

In this phase, the designer needs to clarify what the overall objectives of the target system is, what functional and performance requirement the system should achieve, and so on. This corresponds to making the *enterprise specification* of the target system from the Enterprise Viewpoint (see Fig. 1).

2.2.2 Functional Design Phase

The functional design of the target system can be given by observing it from (1) the information viewpoint, (2) the computational viewpoint and (3) the engineering viewpoint as described below (see Fig. 1).

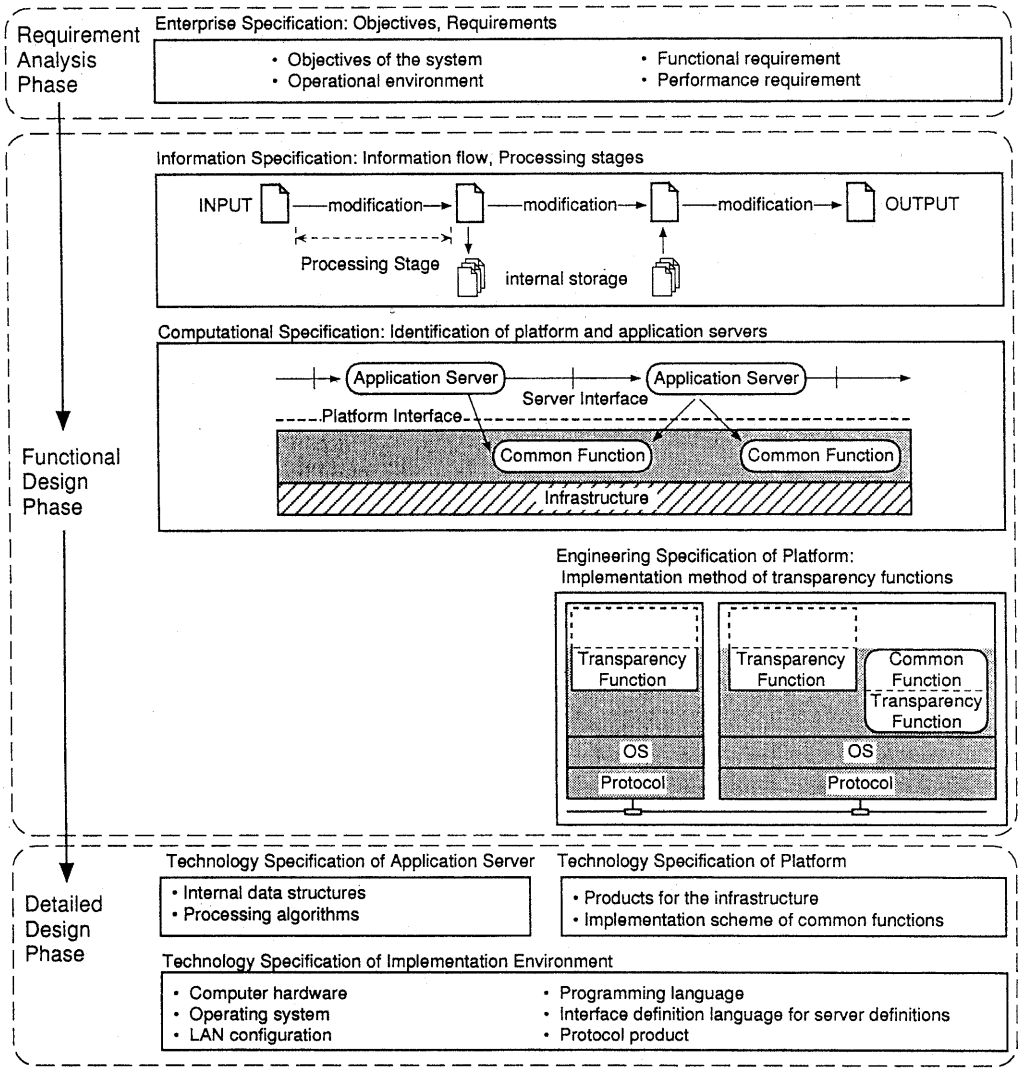

Figure 1: Design Phases and Viewpoint Specifications

(1) The overall behavior of the target system can be described by giving *information specification*. This specification defines the structure of information elements that the target system handles. It also clarifies the information flow within the system by describing which information elements are modified, how those modification are made, which information elements are stored in the system, which information elements are generated in the system, and so on. With this information flow, the system designer can clearly states what the target system does and what *processing stages* the system has.

(2) The functional decomposition of the target system can be done with *computational specification* without paying attention to the distribution of computers. From the computational viewpoint, the target system is divided into the *application servers* which provides the application specific functions, and the platform on which the application servers are built.

The platform is further divided into the *infrastructure* which corresponds to general purpose distribution support environment (DSE) providing the distribution transparency, and the *common functions* which are not provided by DSEs but are commonly used by the application servers. Examples of common functions include log function that holds data in the stable storage.

In the computational specification for the target sys-

tem, the functional modules (the infrastructure, the common functions and the application servers) are identified by considering the requirement from the enterprise specification, the system behavior and the processing stages defined in the information specification. The computational specification also defines the programming interface of each functional module. Those interfaces are defined by referring to the information structure stated in the information specification.

(3) In order to decide how to realize the platform, the designer will describe the *engineering specification* for the platform. Since the platform has the layered structure with the common functions and the infrastructure, this design process will have the following alternatives.

- In the case that the platform can be realized by using an existing general purpose DSE, the engineering specification will not include the internal structure of the platform.
- In the case that some common functions are introduced on the top of an existing infrastructure, the designer need to make the engineering specification describing how the common functions are realized by servers interacting with each other and library routines that are linked with application servers.
- In the case that the infrastructure itself is newly designed for the target system, the engineering specification of the platform describes how the various transparencies are realized in the form of *transparency function* by using the functions provided by the underlying operating system and protocol for the inter-processor communication.

Since the platform provides the distribution support, the application servers do not require any engineering specifications.

2.2.3 Detailed Design Phase

In this phase, the implementation environment, and the detailed design of both the application servers and the platform are described as the *technology specifications* (see Fig. 1). The technology specification of the implementation environment includes: computer hardware and operating system to be used, configuration of local area network, protocol products for the inter-processor communication such as TCP/IP or OSI, programming language, interface definition language for server definitions, and so on.

The technology specification of the application servers clarifies the internal data structures, the processing algorithms that are derived from the operation to the information elements defined in the information specification, and the programming interface defined in the computational specification.

In the technology specification of the platform, the designer investigates if there is a suitable product for the infrastructure referring to the performance requirement in the enterprise specification and the realization scheme defined in the engineering specification for the platform. In the case that the infrastructure itself is to be implemented, the technology specification clarifies the internal structure of the infrastructure using the engineering specification of the platform. The implementation scheme for the common functions should be also described in this specification.

3 Application of Design Method to MHS System

This section shows how the design method described in the previous section can be applied to the implementation of an MHS system as an example of telecommunication system. The MHS system designed here provides an MTA function which relays messages between other MHS systems, and is realized by using multiple computers connected through a local area network (see Fig. 2), We call this implementation a "distributed MHS system" (D-MHS system for short) in the rest of this paper. The following subsections give the specification of the D-MHS system from the corresponding viewpoint.

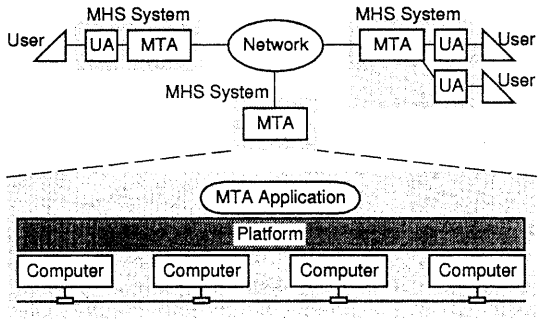

Figure 2: Architecture of D-MHS system

3.1 Enterprise Specification

In the requirement analysis phase, we identified the followings:

- The system should conform to the X.400 protocols.
- The system is built on the multiple computers connected through local area network each other.
- The system should never lose received messages even if the software and hardware crash occurs.
- The system should never duplicate messages accidentally.

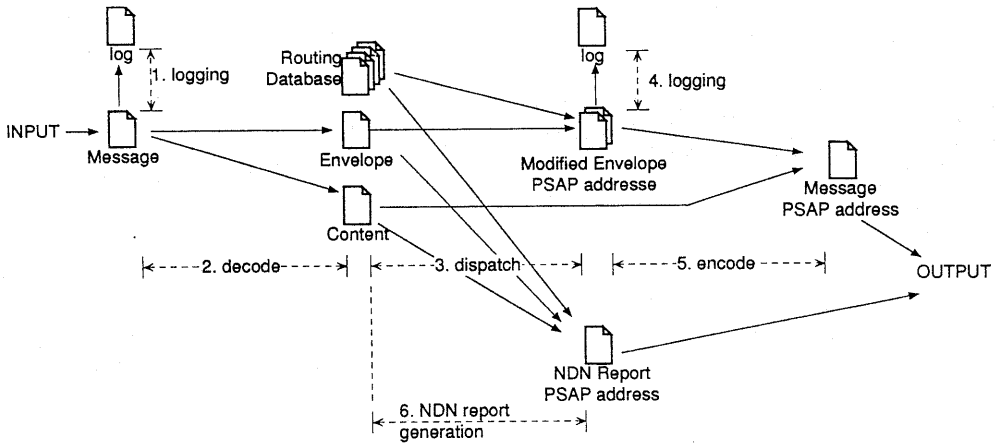

Figure 3: Information Specification of D-MHS system

- The system should support various types of networks between other MHS systems, such as PSPDN (Packet Switched Public Data Network), ISDN (Integrated Services Digital Network), telephone network and leased lines.

3.2 Information Specification

An X.400 MHS system supporting the MTA function handles three types of information: *messages*, *reports* and *probes*. Messages are used to convey the interpersonal message information. A report is generated by an MHS system to notify the originator whether a particular message is delivered to the recipient or not. Probes are used by users to determine whether a particular recipient is reachable or not. The information specification in Fig. 3 shows the information flow for the handling of messages. This specification recognizes the following processing stages.

1. The received message is *logged* into a stable storage in order not to lose it.

2. The D-MHS system *decodes* a message into an *envelope* and a *content*. The envelope is interpreted and modified by the D-MHS system to forward the message, while the content is left untouched.

3. The D-MHS system *dispatches* the message by analyzing the envelope. Among many fields in the envelope, the D-MHS system handles `recipient-name` and `trace-information`.

   (a) The D-MHS system finds the PSAP (Presentation Service Access Point) addresses of the next MHS systems to send this message for each `recipient-name` by making a query to its internal *routing database*.

   (b) If the message needs to be transferred to more than one MHS system, the envelope is replicated. A PSAP address of the next MHS system is associated with each replica.

   (c) The `trace-information` is updated to indicate the action taken by the D-MHS system. It contains the identifier of the D-MHS system, the time when the message is received, the identifier of the next MTA, and the action taken by the D-MHS system (`relayed` or `rerouted`).

4. The modified envelope is logged into the stable storage.

5. The modified envelope and the original content are encoded into the message and sent out to the next MTA.

6. If the route to the recipient of a message is not found, a *non delivery notification* (NDN) report is generated and is sent back to the originator of the message.

3.3 Computational Specification

Figure 4 shows the computational specification of the D-MHS system. We introduced an application server called *MHS server* to realize the processing stages for the message handling identified in the information specification. An MHS server realizes the stages 2, 3, 5 and 6 described in the Fig. 3. An MHS server is assigned to one communication circuit established between other MHS systems. We also introduced a dedicated application server called *routing database server* which is responsible for the translation from `recipient-name` to PSAP address.

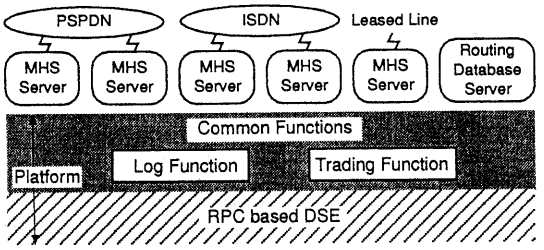

Figure 4: Computational Specification of D-MHS system

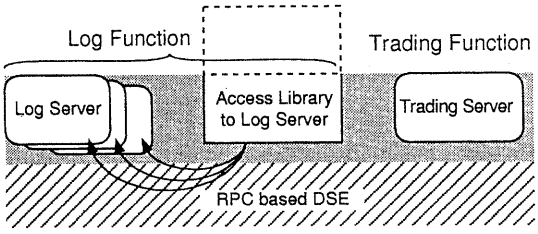

Figure 6: Engineering Specification of Platform

```
// MHS Server -- migrate an MHS message
routine HandleMessage(
    in  envelope : ^char;
    in  content  : ^char;
    in  psap     : psap_t);
// Routing Database Server
// translate a set of OR addresses
// to a set of PSAPs
routine Translate(
    in  recipients : array_of_ORaddr_t;
    out psaps      : array_of_psap_t);
// Log Server
routine Log_Write(
    in  logHeader : log_header_t;
    in  options   : int;
    in  data      : ^char);
```

Figure 5: Examples of Programming Interfaces of Application Servers and Platform

As for the infrastructure of the platform, we adopted the existing distribution support environment (DSE) using the remote procedure call (RPC) based on the client-server model for the ease of implementation.

We decided that *log function* which keeps communication logs safely in the stable storage is supported by the platform as a common function. The log function is implemented by use of multiple *log servers* and provides the followings with the application servers.

- writing and reading capability for a single log,
- *group logging* which allows the application servers to write and read a group of related logs atomically, and
- *log server switching* which detects the failure of log server, find an operational one, and then switches to it.

In the D-MHS system, more than one server with the same type may be replicated for the purpose of the high performance and availability. Therefore, the platform will provide a common function to monitor the load of servers and to choose the most appropriate server to handle a request. This is provided by the *trading function* studied in ODP.

The programming interfaces of the application servers and the platform are also defined in this specification. The examples of those interfaces are shown in Fig. 5 (in the notation of the server interface definition language of the RPC system which we adopt).

3.4 Engineering Specification

Since we adopted an existing RPC based DSE as the infrastructure, the engineering specification of the platform addresses how the common functions are realized on the top of the infrastructure as shown in Fig. 6.

The log function is realized by a number of log servers and a set of access library to a log server. An application server (shown by dotted box in Fig. 6) can use the log function through the access libray to the logserver. The log server accepts `Log_Write()` and `Log_Read()` requests for a single log from clients. The group logging is implemented by an access library using the following scheme. When a group of logs are written, the library marks in the log server that the logs form a group. When a group of logs are read, the library examines whether all the logs in the group are read from the log server. If not, it decides that all the logs in the group were not stored in the log server correctly and it ignores the read logs. The log server switching is implemented by another access library that provides replication transparency for the log servers. When this library detects a particular log server is down by the failure of RPC to that log server, it searches one of the operating log servers and then writes all the internal data to the new log server.

The trading function is realized as a separate server called *trading server*. The trading server inquires the load of each application server and advises other application servers of the least loaded application server. In the design of D-MHS system, the load of an MHS server is measured in the total size of messages waiting to be sent in that MHS server.

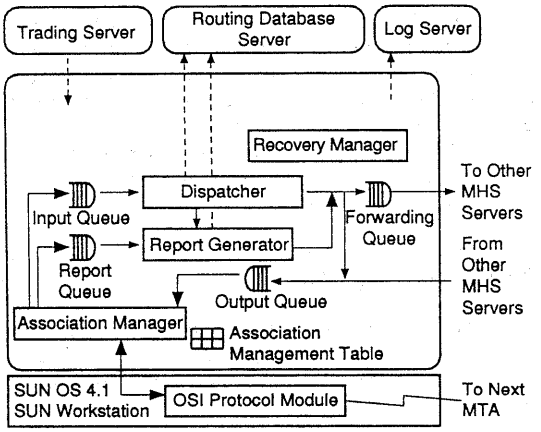

Figure 7: Technology Specification of MHS server

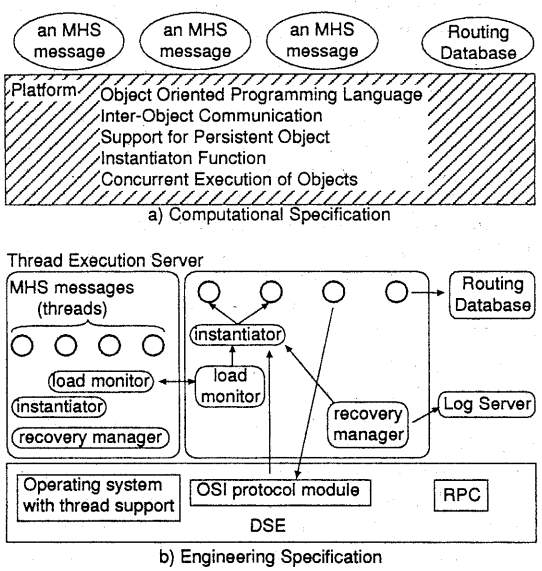

Figure 8: Viewpoint Specifications for Platform based on Object Oriented Distribution Support Environment

3.5 Technology Specification

The technology specification clarifies the implementation environment and the detailed design of the MHS server, the routing database server, the log server, the access library to the log server, and the trading server. This subsection shows the detailed design of the MHS server.

The technology specification of an MHS server was directly derived from the information, computational and engineering specifications of the MHS server described above. Detailed design of the MHS server using actual hardware and software is shown in Fig. 7.

The internal structure is mainly determined from the information specification shown in Fig. 3. The association manager puts the received messages into the input queue. The dispatcher takes one message from the input queue and dispatches it. If the trading server directs the dispatcher to transfer the received message to an other MHS server, it puts the encoded message into the forwarding queue. Otherwise it puts the message into the output queue of itself. NDN reports are generated by the report generator when the route of a message is not found.

Besides the received message and the modified envelope, the MHS server logs the state of each queue and the association management table for recovery from the server crash. In order not to lose or duplicate the messages accidentally, a message must exist exactly one of the input, forwarding, output or report queue (The report queue is a temporary queue for messages that have encountered communication errors). The dispatcher must update these queues in atomically to assure this condition. The group logging provided by the platform is used for this purpose.

4 Discussions

(1) It is considered that the design method described in this paper realizes the systematic design of a distributed system by use of the five ODP viewpoints. The method uses the five viewpoints to specify the target system clearly in the three phases of the top-down design, that is, the requirement analysis, the functional design and the detailed design phase. Especially, it provides a well-organized design process for the functional design phase by applying the information, computational and engineering viewpoints to this design phase. The overall behavior of the target system are clarified by the information specification. The functional modules including the application servers and the platform are identified by the computational specification. The functions of the platform are designed by the engineering specification. As the result, the detailed design phase can be simplified. The detailed design of the application server can be easily described by referring to the information and computational specifications. The detailed design of the platform can be described using the computational and engineering specifications.

(2) The result of applying our method to the D-MHS system indicates that our method will be useful for implementing a telecommunication system as a distributed system. It is because a telecommunication system defines the information flow and the functional modules clearly and therefore the information and computational specifications can be easily described. We believe that the other type of telecommunication system such as the

service control function for the intelligent network [4] can be designed effectively by using our method.

(3) Since our method separates the description of the overall behavior of the target system from the identification of the functional modules, it enables the designer to choose the various implementation scheme. Even after the designer has made the information specification, he can decide what platform is used by taking account of the requirements, the available technologies and so on. Our method covers both cases where an existing distribution support environment is used as shown in the previous section, and where a platform is newly implemented for the target system.

(4) Our method may realize the target system with considerably different architecture, in the case that the different type of platform is selected. The followings shows an example where we adopt the *object oriented distribution support environment* as a platform for the D-MHS system. The enterprise and information specifications are the same as those of the previous D-MHS system. In the computational specification, each MHS message and the routing database acts as an autonomous object on the platform (Fig. 8-a). The platform provides functions for the object class definition, the instantiation of objects, the communication between objects and so on. It also provides support for the *persistent objects* that are durable through system crashes. To realize such a platform, the execution environment for fine-grained objects can be designed using the engineering specification of the platform. In the example shown in Fig. 8-b, an MHS message object is realized as a thread in a thread execution server. RPC is used to realize inter-object communication and the log server is used to support persistent objects.

5 Conclusion

In this paper, we described a top-down design method based on the ODP viewpoint approach for a distributed system which is built on multiple computers connected through high speed networks. The method uses the five viewpoints to specify the target system clearly in the three design phases, that is, the requirement analysis phase, the functional design phase and the detailed design phase. The specification from the enterprise viewpoint is used in the requirement analysis phase. The specifications from the information, computational and engineering viewpoints correspond to the functional design phase. The functional design includes the design of the platform and the application servers realized on top of the platform. The specification from the technology viewpoint forms the detailed design of both the platform and the application servers. The method provides a well-organized design process for the functional design phase, and as the result the detailed design phase can be simplified and the technology specification can be described by referring to the information, computational and engineering specifications.

The design method was examined by applying to the design of a distributed MHS system supporting X.400 protocols. In this example, the RPC based on the client-server model was selected as the base of the platform, and the log function and the trading function are realized in the platform as the common functions. The result showed that our method can simplify the complexity of the design for a distributed telecommunication system by introducing the systematic design approach.

Acknowledgement

The authors would like to thank Dr. Y. Urano, director, and Mr. K. Maya, deputy director of KDD R & D Laboratories, for their continuous encouragement of this study.

References

[1] ISO, : *Information Technology – Basic Reference Model of Open Distributed Processing – Part 2: Descriptive Model*, ISO/IEC JTC1/SC21 (1992).

[2] ISO, : *Information Technology – Basic Reference Model of Open Distributed Processing – Part 3: Prescriptive Model*, ISO/IEC JTC1/SC21 (1992).

[3] ISO, : *Draft Recommendation X.901: Basic Reference Model of Open Distributed Processing – Part 1: Overview and guide to use*, ISO/IEC JTC1/SC21/WG7 (1993).

[4] Kato, T., Fujinaga, M. and Suzuki, K.: Applying Distributed Processing Technologies to Intelligent Network, *IEICE Transactions*, Vol. E 74, No. 11, pp. 3672 – 3682 (1991).

[5] Geihs, K. and Mann, A.: ODP Viewpoints of IBCN Service Management, Technical Report 43.9104, IBM European Networking Center.

[6] Fleck, J. J., Liou, C. C., Natarajan, N. and Phillips, W. C.: The INA Architecture: An Architecture for Information Netwroks, in *Proceedings of the TINA 92*, No. 13-1 (1992).

[7] Brégant, G.: Platform Modelling Requirements from the ROSA Project, in *Proceedings of the TINA 92*, No. 11-1 (1992).

[8] Fujinaga, M., Kato, T. and Suzuki, K.: Implementing IN Functional Entities on top of Distributed Operating System, in *Proceedings of the XIV International Switching Symposium*, Vol. 1, pp. 268 – 272 (1992).