Object-oriented design of a Message Handling System protocol.

S.Erradey * M.Kadoch ** G.V.Bochmann*

* Informatique et de recherche opérationnelle.
Université de Montréal
Montréal, Québec
Canada.

**Département de génie électrique
École de technologie supérieure
Montréal, Québec
Canada

ABSTRACT - This paper presents a specification and object-oriented modelling technique for a distributed application. The object-oriented specification of a distributed application provides a natural mapping of the system components onto objects and supports the implementation of reusable software components. The application is the X.400 Message Handling System defined in its 1988 version in the context of the Reference Model for Open Systems Interconnection. The BOOCH object-oriented analysis and design method has been used to facilitate the mastery of the notation and the process of object-oriented design by understanding the fundamental elements of the object model such as abstraction, encapsulation, modularity, hierarchy, typing, concurrency and persistence.

I. INTRODUCTION

A common goal in developing a distributed application is how to provide a suitable information model to capture the essential structure of information components so as to facilitate the design and implementation process. To reach this goal, a precise definition of the information components and their interaction with each other is necessary. To cope with the complexity inherent to distributed applications, abstraction, encapsulation, modularity, and hierarchy are the concepts that prevails.

These are the basic principles of object-oriented model. Thus the specification of a distributed application can be modelled as a collection of objects mapped onto the information components. These objects interact with each other following the interface specification of the components.

To master the process and the notation of the object-oriented design, the BOOCH method [1] has been used. Indeed this method provides a good framework for the design of object-oriented applications of different types. Yet the use of this method for distributed applications had to be experienced.

The Message Handling System (MHS) [2] is the distributed application used to show the design process of such applications using object-oriented modelling in general and the BOOCH method specifically. But the technique is applicable to other distributed applications developed in the OSI environment.

The paper is organized as follows : Section 2 defines the structure of the application layer as presented in the OSI reference model. Section 3 describes the MHS, its functional model, protocols and services as presented in the 1988 version of the X.400 series. Section 4 gives a brief introduction of the object-oriented model as defined by BOOCH. Section 5 features the object-oriented realization of the protocol. The conclusion gives an evaluation of the experience.

II. THE APPLICATION LAYER STRUCTURE

In an effort to normalize the growing number of communication protocols, the ISO and the ITU-T (formerly CCITT) joined to define the Basic Reference Model for Open Systems Interconnection (OSI) [3]. These recommandations are aimed to regulate the communication between heterogeneous systems. The communication functions are represented into seven layers. Each of these layers provides a specific service to the upper layer and uses the service of the lower layer. The layers one to four provide transport-oriented services and the layers five to seven provide application-oriented services.

The application layer [4] (layer 7) is the interface between the communication system and the application process (AP) which is the abstract representation of a program performing information processing in a distributed environment. Since it is the upper layer, there is no application access point and no application connection. An AP accesses the OSI services through application entities (AE). Every AE has access to the presentation layer through one or more presentation service access points (PSAP), thus providing them with an address in the OSI environment. Each AE is assigned to an AP and can therefore exist only if the AP is activated. In order for an AP to communicate with its remote peer, it uses one or more AEs associated with AEs in the remote machine. This link is called application association (AA).

The application layer functions are regrouped into different application service elements (ASE). The conceptual schema that defines the use of ASE and their interaction within an AE is called the application context (AC). The ASE can be subdivided in two groups : the common ASE, independent from the application, such as ROSE, ACSE, RTSE,...; and the specific ASE part of a particular OSI application, such as the MSSE, MDSE, MASE specific to the MHS application.

In the AE, the coordination between the different ASE is realized by the Single Association Control Function (SACF). The set of functions and information of a particular association are regrouped in the Single Association Object (SAO). The SAO always holds an Application Control Service Element (ACSE), a SACF function and other ASEs. An AE can communicate simultaneously with different remote AEs. This means that an AE has to manage more than one association, therefore more than one SAOs. A specific function called the Multiple Association Control Function MACF is used to coordinate the different SAOs. The AP and AE represent a set of resources forming static objects. A particular execution of these objects is viewed as an instantiation of the object called, respectively, AP and AE invocations.

III. THE MESSAGE HANDLING SYSTEM X.400

The electronic mail is probably the most popular application of the last ten years. The multiplication of private E-mail systems pushed the ITU-T to standardize it under the X.400 series in 1984. In 1988, with the OSI environment standard, the ITU-T and ISO joined to integrate the X.400 as an OSI application. This application is known in either names, the Message Handling System MHS [2] or the Message Oriented Text Interchange System MOTIS.

The primary function of a mail system is to provide the means to send and receive messages. Users, that is originators, access the MHS system by sending their messages through a user agent (UA). The message is either submitted to the Message Transfer System (MTS) or stored in the Message Store (MS) for an ulterior submission. The MTS then routes the message through the Message Transfer Agents (MTA) and delivers it to one or more recipient UA or MS. The MTS can return notifications to the originator. From the UA, the message is delivered to the recipient.

Four protocols exist in the MHS to provide the access to the different services and functions. This paper focuses on one of them, namely the MTS-access protocol, known as P3. This protocol provides the UA or MS (MTS-users) with the operations [5] to access the MTS.

The MTS and its users communicate through ports : submission, delivery and administration ports. Each of these ports supports a set of remote operations. Through the submission port, the MTS-user submits a message (Message-submission), tests the submission and delivery of a message (Probe-submission) and may suspend the delivery of a submitted message (Cancel-deferred-delivery). The MTS can impose constraints on the use of the previous operations (Submission-control).

Through the delivery port, the MTS delivers a message to the MTS-user, acknowledges the MTS-user of one or more outcomes of a previous invocation of a Message-submission or Probe-submission operation. The MTS-user can impose constraints on the use of delivery port operations by the MTS.

Through the administration port, the MTS-user makes long-term changes to various parameters held by the MTS concerned with the delivery of messages to itself (Register) and changes its credentials held by the MTS. The MTS can also change its credentials held by the MTS-user.

Before the MTS and its user can communicate and call abstract operations upon each other, they must bind. The MTS-bind enables the MTS-user to establish an association with the MTS, or the MTS to establish an association with an MTS-user. The MTS-unbind enables the release of an established association by the initiator of the association.

In the P3 protocol [6], an OSI environment protocol, the MTS and its user represents application processes (AP) located in different open systems. As seen above, the communication between two APs is materialized between an association of two application entities (AE) using the presentation service. The functions of the AE are supported by the application service elements (ASE).

The access to the MTS abstract service is taken under the charge of three ASE, each one functioning under the Client/Server paradigm between the MTS-user and the MTS. The Message Submission Service Element (MSSE), the Message Delivery Service Element (MDSE) and the Message Administration Service Element (MASE) support respectively the submission, delivery and administration ports. These specific ASE are supported by common ASEs such as the Remote Operations Service Element (ROSE), the Association Control Service Element (ACSE) and the Reliable Transfer Service Element (RTSE).

The use of the last ASE depends on the application context supported in a given association. Indeed the RTSE is needed when a reliable transfer of the messages is not guaranteed. The use of RTSE or not and the initiation of the association by the MTS-user or the MTS create four different application contexts for the P3 protocol.

The application contexts, the abstract operations and the Application Protocol Data Units APDU exchanged are defined using the Abstract Syntax Notation One (ASN.1) [7]. This notation specifies also the transfer syntax of data.

IV. THE BOOCH OBJECT-ORIENTED METHOD

The BOOCH method is an Object-Oriented Analysis and Design OOAD method defined by Grady Booch. It encompasses both a process and a notation. The process represents the steps by which a developer passes to reach a good design in conformance with the system requirements. In the Booch method, this is viewed as two processes interacting with each other : the macro-process and the micro-process.

The macro-process is inspired from the standard Top-Down model of development (Cascade). It includes the following phases : system requirements, analysis, design, implementation and the maintenance. In the field of communication protocols, the live cycle of a protocol is as follows : system requirements, analysis (informal specification), formal specification, validation, implementation, testing and conformance testing.

The second process known as the micro-process is inspired by the Boehm spiral model. The micro-process includes the following activities :
- Identification of classes and objects.
- Identification of the semantics of classes and objects (behaviour of classes).
- Identification of relations between classes and objects.
- Implementation of the classes and objects.

The specification or the implementation phase of a communication protocol using the BOOCH method, can be viewed as the application of the micro-process during each of the two phases.

To design a system, we need a notation describing every detail and views of the designed system. The BOOCH notation [8] [9] is subdivided in four models : a logical model, a physical model, a static model and a dynamic model.

The logical model describes the existence and signification of key abstractions and mechanisms that constitutes the problem domain or the system architecture. To illustrate this view, the class diagram and object diagram are used. The class diagram shows the existence of classes and their relationships. The object diagram, known also as the scenario diagram, shows the existence of objects and their relationships. It also shows how the scenarios work on the objects and their relationships (exchanging messages).

The physical model shows the concrete software and material composition of the system and its implementation. The module and the process diagrams illustrate this model. The module diagram shows the distribution of classes and objects into modules. Whereas, the process diagram shows the attribution of processes to the processors.

In object-oriented design, we express the dynamic semantics of the architecture and its implementation with two diagrams. First, the state transition diagram shows the state space of an instance of a given class, the events that triggers the passage from a state to another and the actions that result. Second, the interaction diagram shows the execution of a scenario in the context of an object diagram.

V. OBJECT-ORIENTED SPECIFICATION OF THE P3 PROTOCOL

In the section III, we explained how the P3 protocol, being an OSI application protocol, is structured and uses the application service elements (ASE) and the presentation service. The MTS and the MTS-user are application processes (AP). Instances of this AP called API represents the execution of a particular function of the MTS or MTS-user. The MTS and the MTS-user represent a class abstraction called the AP class (Fig. 1). The API are considered objects of the AP class. The MTS and MTS-user communicate through three ports (submission, delivery and administration). The three ports constitutes three different classes. The relation between the AP class and the port classes are a 'has' relationship. The AP uses the OSI communication services through application entities (AE).

The AP may need more than one AE. Indeed each AE instance will be supporting a particular application context (AC). The AE represents a class abstraction. The relationship between the AP class and the AE class is a 1-to-n 'use' relationship. For each of the AE associated with a remote AE, there is a single association object. The SAO abstraction is represented in a class. The relationship between the AE and the SAO is a 1-to-n 'has' relationship. The SAO holds several ASE. Each ASE is represented by a class.

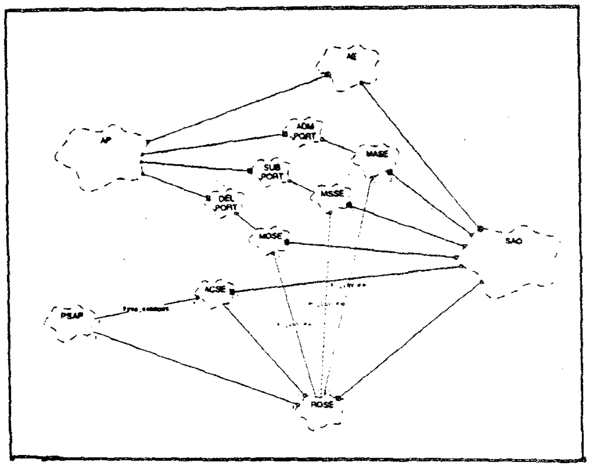

Fig. 1 MHS Class Diagram.

In the AC supported by our specification, the following ASE are present : ACSE, ROSE, MSSE, MDSE, MASE. The relationship between the SAO and the different ASE are a 1-to-1 'has' relationship. Every AE has access to the presentation layer through one or more presentation service access point (PSAP). The PSAP is represented by a class. The relationship between the AE and the PSAP is a 1-to-n 'use' relationship.

The specification of a class holds different items such as :
Class name : ACSE
Documentation : Association control service element, establishes, releases and aborts application associations.
Export Control : Public
Cardinality : 1
Hierarchy :
Superclasses : none
Public uses : PSAP Pres_connect
Public Interface :
Operations :
AASC.req, ARLS.req, AASC.conf, AASC.resp, AASC.ind, ARLS.ind, ARLS.resp, ARLS.conf

State machine : Yes
Concurrency : Active
Persistence : Transient

The operations of ACSE are the primitives that accesses the service according to the standard. Export Public specifies if the class is imported from another class diagram and private specifies if it is exported to another class diagram. Cardinality specifies the number of instances of this class that could be created. Hierarchy indicates if the current class is inherited from a superclass. A 'use' relationship exists between ACSE and PSAP. The relation is called 'Pres_connect'. Concurrency indicates if the instances of this class are concurrently active. Persistence shows if the existence of the class is time independent.

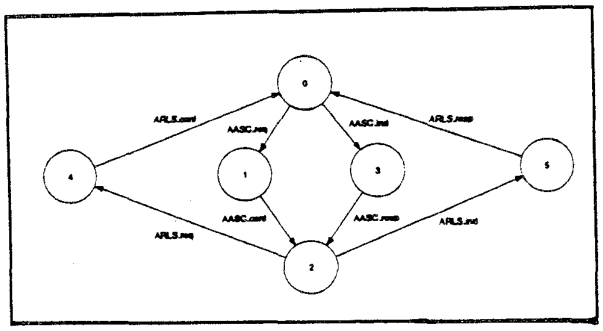

Fig. 2 ACSE state diagram

As indicated in the specification, the ACSE class has a state machine. The best way to document the dynamic behaviour of certain classes is to use the state machine diagram. This is specially true for OSI protocol specifications since it is currently used as a formalism in many of its standards. The diagram shows the state space of a class, the events that cause a transition from one state to another and the actions that result. For the ACSE state diagram (Fig. 2), The state 0 represents the state idle, state 2 the state associated in the ACSE protocol. The incoming events and the outgoing events represents calls to the ACSE operations.

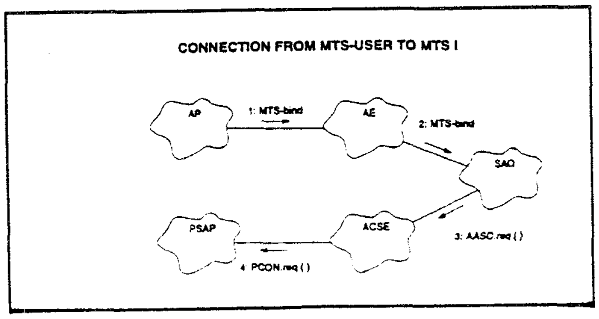

Fig. 3 BIND scenario diagram

We use object diagram (scenario) to show the behaviour of the system in different cases. The diagram shows objects, relationships and exchange of messages using numbers to notify the order of the operations. For example, the creation of a bind between the MTS-user and the MTS. This scenario diagram (Fig. 3) shows how the MTS-bind operation is initiated by the AP. The AE notifies the SAO which in turn activates the ACSE object calling the AASC.req operation. The ACSE translates that into a PCON.req call to the PSAP.

VI. CONCLUSION

In the OSI environment, the application-oriented layers present a great deal of complexity in terms of information components. The MHS is no exception to that. The use of object-oriented technology is easily justified by the fact that supported concepts like abstraction, encapsulation, modularity and hierarchy help deal with this complexity. In addition, the capability of the BOOCH method to master a notation and a process, both important aspects of object-oriented analysis and design, makes the whole process a lot easier. Further, the Rational ROSE tool supporting the BOOCH method offers a partial code generation function that helps in the implementation process. Nevertheless, there is a great need for a verification tool that could verify the semantics of the model and help eliminate certain types of errors in the earlier stages of the analysis and the design.

VII. ACKNOWLEDGEMENT

This work has been partially funded by the Synergie project IGLOO.

VIII. REFERENCES

[1] G. Booch, Object-oriented analysis and design with applications, Benjamin/Cummings, 1991.

[2] ITU-T, X.400, CCITT recommandation, Message Handling System : System and service overview, ITU-T, 1988.

[3] ITU-T, X.200, CCITT recommandation, Open Systems Interconnection Reference Model, ITU-T, 1988.

[4] ITU-T, X.207, CCITT recommandation, Application layer structure, ITU-T, 1988.

[5] ITU-T, X.411, CCITT recommandation, Message Transfer System : Definition of abstract services and procedures, ITU-T, 1988.

[6] ITU-T, X.419, CCITT recommandation, Protocol specification, ITU-T, 1988.

[7] ITU-T, X.208, CCITT recommandation, Abstract Syntax notation One specification, ITU-T, 1988.

[8] G. Booch, The Booch method : notation part I, Computer language, pages 47-70, September 1992.

[9] G. Booch, The Booch method : notation part II, Computer language, pages 37-55, October 1992.