A Critical Analysis of the X.400 Model of Message Handling Systems

Marten van SINDEREN
and Evert DORREGEEST

Twente University, PO BOX 217, 7500 AE Enschede, The Netherlands, uucp: mcvax!utinul!sinderen

The CCITT X.400 model of store and forward Message Handling Systems (MHS) serves as a common basis for the definition of electronic mail services and protocols both within CCITT and ISO. This paper presents an analysis of this model and its related recommendations from two perspectives. First the concepts of service, protocol and interface are discussed together with their application to this model; second the positioning within ISO's reference model for Open Systems Interconnection (OSI) is commented on.

Keywords: X.400, Message handling systems, Open systems, Structuring concepts.

Marten J. van Sinderen received his M.S. degree in electrical engineering in 1982 from the Twente University, Enschede, The Netherlands. Since 1982, he has been a member of the Department of Computer Science of the Twente University, working on the functional design of distributed computer systems. His research interests include protocol specification, the use of formal description techniques, network interconnection, and higher level protocols. Van Sinderen is an active contributer to ISO/TC97/SC21/WG6 and WG5 on Open Systems Interconnection.

Evert Dorregeest studied at the Twente University in Enschede, The Netherlands, from 1982 to 1987, obtaining an M.S. degree in computer science. He is presently working in military service at the military computer centre in Apeldoorn, The Netherlands. His research interests include computer networks and electronic mail systems.

North-Holland
Computer Standards & Interfaces 7 (1988) 363–375

1. Introduction

The major impetus for the development of electronic mail systems has been provided by office automation applications. In office environments, electronic mail systems facilitate interpersonal message exchange, both from originator to single recipient and from originator to multiple recipients. Optimal usage in such an environment requires that an electronic mail system should be able to accept messages from a number of information sources (serving the originator) and support the delivery of messages to a variety of information sinks (serving the recipient). Messages are then not restricted to simple text but may contain various information types such as voice, facsimile and graphics. Also, submission and delivery of messages may either be interactive or spooled depending on the mixture of source and sinks.

A number of electronic mail systems have already been implemented. They have, however, often a limited application, being closed corporate systems (DECnet), part of a research network (EARN/BITNET, JANET), vendor-specific, or aimed at single system communities (EUNET/USENET) [4] discusses several such systems and their limitations). The comfort gained through an electronic mail system would be greatly enhanced when the system in not limited to the premises of an organization or constrained by specific implementations.

These user needs, as well as the potential market, are recognized by the CCITT, ISO and ECMA. They are currently making considerable efforts to define office document architectures, office document interchange formats [5], and services and protocols for message handling. These definitions are *abstract* in the sense that they do not rely on any specific coding or system implementation. In this paper we will analyse the message handling services and protocols as defined by CCITT in their X.400 recommendations for Message Handling Systems (MHS) [7]. MHS has gained broad acceptance among user communities and computer manufacturers, and is used as the basis

for ISO's Message Oriented Text Interchange System (MOTIS) [8].

The paper is organized as follows: Section 2 presents the basis architecture for MHS and summarizes the message transfer facilities which can be offered. Section 3 is a short tutorial on the concepts of service, protocol and interface as expedients for structuring communications systems. In Section 4, MHS is explained in more detail and the application of the structuring concepts is analyzed. The discussion is limited to communication aspects of MHS; for example, aspects of authentication, access restrictions and naming directories are not covered here. Section 5 analyses whether the proposed placement of MHS in the Open Systems Interconnection (OSI) reference model, namely on top of the OSI presentation service, yields an economical design. Section 6, finally, summarizes our findings from the previous two sections and presents some concluding remarks.

2. Summary of X.400

2.1 Layering

The X.400 architectural model has two layers (*Fig. 1*). The lower layer is the Message Transfer layer, which is made up of *Message Transfer Agent Entities* (MTAE) and *Submission and Delivery Entities* (SDE). The protocol which governs an MTAE (the communication between two MTAEs) is the message transfer protocol (P1). This protocol is concerned with the store-and-forward transfer of messages. That is, messages are sent from one end- or intermediate system MTAE to another end- or intermediate system MTAE. MTAEs provide storage of messages and can perform certain manipulative actions on them according to their included protocol control information. Forwarding a message may also imply sending it to a number of subsequent MTAEs, instead of one, in order to offer multi-recipient delivery. The protocol governing

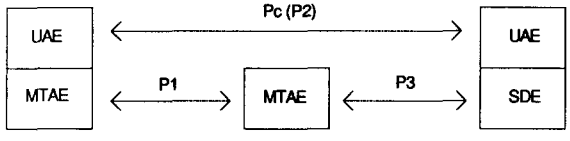

Fig. 1. Layered model of MHS.

an SDE (the communication between an SDE and a MTAE) is the submission and delivery protocol (P3). P3 primarily provides a reliable exchange of messages and does not support particular end-to-end electronic mail functions; the messages exchanges are therefore "simple" messages, i.e. they contain the user-supplied information but not the additional protocol control information used, and generated by, P1. P3 is used to provide a distant application process with access to the message transfer functions.

The P1 and P3 protocol are both based on the OSI presentation service. Their coordinated operation provides the message transfer service which is available to the entities in the upper layer.

The upper layer is the User Agent layer, and consists of *User Agent Entities* (UAE). A range of protocols (Pc) can be defined at this level, each of them concerned with a particular syntax and semantics of data which is transparently transferred via the message transfer service. To date, only the interpersonal messaging protocol (P2) is defined. As the name suggests, this protocol supports the electronic equivalent of paper-based mail (memo) exchange between human participants.

2.2 Message Transfer Facilities

The message transfer service enables a UAE to submit messages destined to one or more recipient UAEs. If a message cannot be delivered, the originating UAE will normally be informed about this fact. The service is not connection-oriented: submission of data takes place without any previous interaction with the other side being required. The message transfer protocol can perform the following functions, among others, on request of an originating UAE:

1. notification of successful delivery of a message, or prevention of notification in case of non-delivery;
2. conversion of the encoded information type (see *Note*) on a message as specified by the UAE, or prevention of any conversion (otherwise, the message transfer protocol may optionally perform type conversions to enable delivery of a message);

   Note: An encoded information type is a particular encoding for instances of an abstract data type defined, or implied, by an application (e.g. codings used for

telex, teletex, videotex, facsimile, document interchange, etc.). The conversion mentioned thus concerns a conversion between codings of instances of possibly different, but "close", abstract data types (in case of different abstract data types loss of information may occur).

3. deferring delivery of a message until a specified date and time has elapsed;
4. returning the content of a submitted message to the originator in case it could not be delivered;
5. performing the transfer of a message in an urgent or non-urgent fashion;
6. disclosure of other recipients to each recipient UAE upon delivery of a multi-recipient message;
7. delivery of a message to an alternate recipient when the actual recipient UAE is not accessible;
8. probing the transfer and delivery of a (pseudo-) message as specified by the UAE.

In addition a recipient UAE can request:

9. holding messages destined to it, thus deferring their delivery, on certain specified criteria.

The primitives and some of their associated parameters, by which the above facilities can be requested, are discussed in Section 4, together with the supporting protocol structures and elements.

3. Architectural Concepts for Structuring a Communication System

Layering is one of the basic structuring techniques used in describing the communication functionality in distributed systems. It is applied in all modern network architectures to control their complexity and to achieve independency of logically unrelated functions. Also the MHS model makes use of this structuring technique.

Layering is based on the concepts of service, protocol and interface. Since we base our analysis of MHS on these concepts, we need a common understanding of them. The following descriptions are believed to be in line with the OSI reference model [9,10].

3.1 Service

Peer users of a distributed system communicate with each other by using their common intermediate – the distributed system – according to certain strict rules. This usage consists of different types of interactions between a user and the underlying system during which parameter values are established to which both the user and the system can refer. The elementary interactions (service primitives) possible between a user (service user) and the distributed system (service provider), their relevant parameters, and their relation to any other such interactions are defined by a *service*.

A service defines the external view of a system, as can be observed by its users. Actually, this *observational behaviour* is what really matters to the users: to define further interactions on top of the system they need not know the internal structuring and functional complexity of the underlying system. The definition and representation of service primitives should be consistent with this view; thus:

- a service primitive expresses useful interactions in the light of communication (i.e. interactions with only local repercussions should be omitted in a service definition). Note that spontaneous actions internal to the provider may also result in the execution of service primitives;
- the parameters of a primitive indicate what is relevant for both user and provider; information only relevant to the service users is transferred in a "transparent" data parameter.

The boundary between a service provider and a service user, where they can execute primitives is called a service access point (SAP). Since this boundary is a conceptual one and may be internal to a real world system, service primitives must be defined in such a way that their implementation is not constrained. This means that their definition is at a *high(est) level of abstraction*.

A more profound discussion of the service concept and its importance in the design of protocols can be found in [12].

3.2 Protocol

As mentioned above, a service does not define how some externally observable behaviour is achieved. This is defined by a protocol. A *protocol* defines the rules for exchanging and manipulating messages (protocol data units, PDUs), with an agreed format and coding for control information, between protocol entities; not to forget, it also relates the service primitives with the PDUs to make the external effects of its functioning clear.

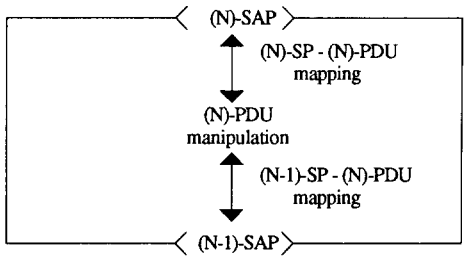

Fig. 2. Representation of a (N)-protocol entity (SP: service primitive; PDU: protocol data unit).

Service and protocol definitions can be applied *iteratively* to the design of distributed systems, as is illustrated by the OSI model. In a layered architecture an (N)-protocol is based on an (N-1)-service, and their composition provides a behaviour equal to that defined by an (N)-service. In this case, the protocol defines as well the relation between its PDUs and the service primitives of the underlying service.

Fig. 2 shows the representation of an (N)-protocol entity as an abstract machine performing mappings and manipulations according to the (N)-protocol. From the service discussion we know that an (N)-PDU is always represented in a data parameter of an (N-1)-service primitive, since its interpretation should be restricted to the (N or higher level)-protocol entities.

3.3 Interface

The local ordering of service primitives at a SAP and the interdependencies between, and restrictions on, their parameter values are described by an *abstract interface* (an abstract interface definition is therefore part of a service definition). An interpretation which is more often associated with the term interface is that of an *implementation description* of an abstract interface; we call this a *real interface*. In designing the real interface between a user and its service provider it may well turn out that the physical distance between the two causes such problems that further protocol engineering is required. The service and protocol concepts can then again be used for structuring purposes; in fact they can be *recursively* applied at different levels of abstraction. In this case, recursive application to an abstract interface yields a set of "interface" services and "interface" protocols.

4. X.400 Services and Protocols

We will now return to MHS. It is our objective to analyse the modeling of electronic mail functions in MHS and to investigate to what extent the X.400 recommendations are suitable prescriptions for "open systems interconnection". The latter means that we demand a general-purpose, *implementation-independent*, description, which leaves implementation freedom where possible and restricts implementations where necessary to allow interconnection and interworking of heterogeneous systems.

4.1 Message Transfer Layer (X.410, X.411)

Table 1 lists all primitives which have been defined for the message transfer service. The primitives are grouped on basis of their partake in certain activities. We can observe that some activities are *local*, i.e. they do not involve interactions which are *remote* to the initiator of the activity.

Non-local, or global, activities involve two or more users in different systems, and imply the coordinated behaviour of these users. The minimum coordination is defined by the service which is provided by the underlying distributed system. Local activity involves only one user (and the underlying system); there is no need for coordination, according to some service definition, with another user. In *Table 1* only "transfer" is considered as a global activity. The "transfer" primitives are therefore the relevant service primitives for the message transfer service, discussed in Section 4.1.1. Section 4.1.2 discusses the message transfer protocol, restricted to the support of the "transfer" interactivity.

"Local" and "global" are, of course, relative notions. We can take a closer look at a local interactivity and may find that this, too, involves several distinguishable entities (e.g. representing a workstation, channel and host) whose interactions can be described in terms of service and protocols, thus introducing a new level of locality. In Section 4.2.3 we will discuss the message transfer interface, where we consider the other activities mentioned in *Table 1*, but also reconsider the "transfer" activity.

Standardizing the local activities of *Table 1* is useful when a user agent and its message transfer agent fall under different implementation authori-

Table 1
Message transfer service primitives in X.411 (req = request, ind = indication, rsp = response, cnf = confirmation).

Primitive	Types	Function
transfer:		
SUBMIT	req, cnf	submission of message
DELIVER	ind	delivery of message
PROBE	req, cnf	submission of probe
NOTIFY	ind	notification of (non) delivery of message or result of probe
local logon / logoff:		
(UAL)LOGON	req, cnf	user logon to system
(MTL)LOGON	ind, rsp	system logon to user
LOGOFF	req, cnf	logoff by user
access management:		
(UAL)CHANGE-PASSWORD	req, cnf	change of user's password
(MTL)CHANGE-PASSWORD	ind, rsp	change of system's password
transfer restrictions management:		
REGISTER	req, cnf	registration of user's receipt restrictions
(UAL)CONTROL	req, cnf	change of receipt restrictions
(MTL)CONTROL	ind, rsp	change of system's acceptance restrictions
local transfer annul:		
CANCEL	req, cnf	cancel request for submitted message

ties and are physically separated. In the X.400 recommendations it is recognized that a user-implemented UAE can be incorporated in a stand-alone workstation which must then interwork via an administration-supplied MTAE. This led to the definition of a separate protocol, the submission and delivery protocol. The definition of the distributed interface primitives and those of the end-to-end service are distinguished here, contrary to the X.411 recommendation, since they concern *different levels* of abstraction.

4.1.1 Message Transfer Service

The message transfer service enables the transfer of messages and probing the transfer of messages, as illustrated by the simplified time diagrams in *Fig. 3.*

Submission of a message is initiated by a SUBMIT request and is locally confirmed by a SUBMIT confirmation. Facilities (1) through (7), listed in Section 2.2, can be requested in the SUBMIT request by setting appropriate parameters. (Some of these facilities are essential – they must be provided when requested – while others are additional – they may be ignored by the system). Provided that the SUBMIT confirmation indicated "success", zero, one or more deliveries may occur by means of DELIVER indications. Depending on the requested facilities, the originating user agent may be informed of successful or unsuccessful deliveries by means of NOTIFY indications.

Probing whether a specified message can be delivered to one or more user agents, is requested in a PROBE request. Again, this request is locally confirmed. The result of this request will be reported back to the originating user agent in one or more NOTIFY indications.

A NOTIFY indication may report on several (would-be) deliveries of a single issued (pseudo-) message. This is only possible when the reports were generated by the same MTAE and the same type conversions were performed on each of the associated message copies.

Analysis: The following comments can be made w.r.t. the message transfer service description in X.411:

- the SUBMIT handshake is described with unnecessary detail; it can be represented as a *single* abstract interaction without degrading the service definition. This comment needs some further explanation.

  The SUBMIT confirmation seems to be intro-

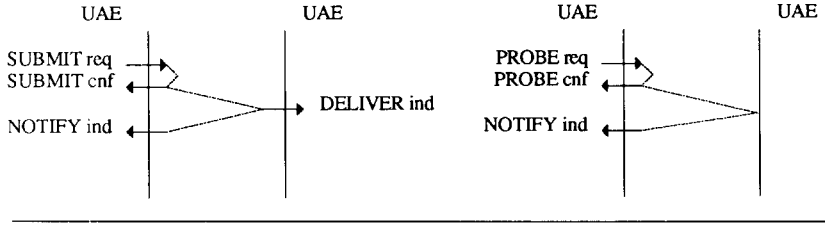

Fig. 3. Time sequence diagrams for transferring a message (with notification of delivery) and probing the transfer of a message. Only one recipient is shown.

duced for two reasons:

1. it takes into account the fact that implementations are subject to failures and represent finite capacities; therefore, a local confirmation of a submitted request can be used to provide certainty about the acceptance of the request;
2. it is used to define a flow of information which is from provider to user, as opposed to that in the corresponding request.

We recall, however, that a service primitive should be defined at the highest possible level of abstraction, not showing details which have only local relevance. Further, the direction associated with a primitive merely indicates the main flow of information [10]; parameter values associated with a primitive may be passed in either direction as appropriate for the primitive. A request for a service which is not acceptable for some local reason is considered as an unsuccessful interaction; such interactions should *not be visible* in the service. Once all parameter values have been established in a primitive execution, the primitive has completed successfully. After this, the provider may report on its inability of progressing the request or on the successful performance of the requested service. Both aspects are already modeled by the NOTIFY primitive.

- the PROBE handshake can be *omitted* completely in the service definition. The reason for this is that a PROBE request will never cause any interactions with a remote user agent, hence there is no need for coordination between users. On the other hand, interworking of MTAEs is required for fulfilling such a request. A protocol element defining this interworking can be considered as part of a management protocol; accessing its functions is a local matter.
- the *relation* between primitives (as in *Figure 3*) is poorly described in the service definition. Although this relation can easily be derived in this case, making it explicit in the service is generally useful to get a quick understanding of the externally visible effects of the service provider. For example, it would have shown which primitives have remote effects and which have not, and how the provider may influence the remote effects (loss of data, manipulation of parameters). For a full understanding of the relation between message transfer primitives we are now obliged to study both the message transfer protocol and the presentation service.

4.1.2 Message Transfer Protocol

An MTAE executing the message transfer protocol is modeled as consisting of three subentities: the message dispatcher, the association manager, and the reliable transfer server. The *message dispatcher* performs the relaying of messages, generation and forwarding of delivery reports, and information type conversion. The *association manager* controls the establisment and release of associations between MTAEs. The role of the *reliable transfer server* (RTS) is to provide and maintain the associations requested by the association manager, to release them when requested, and to perform the transferring of PDUs on basis of available associations.

The service primitives and PDUs which are used by these subentities are shown in *Table 2*. The association manager employs only the OPEN and CLOSE primitives for requesting a new or releasing an existing association, on basis of local management information; PDUs are not defined for these purposes. The message dispatcher employs two types of PDUs: the *user* MPDU, carrying a message submitted by a user agent for delivery, and the *service* MPDU which carries either a probe or a delivery report (MPDU stands for message PDU). MPDUs are mapped onto the user data parameter

Table 2
Service primitives and PDUs used by the association manager, message dispatcher and reliable transfer server.

Association manager and message dispatcher			Reliable transfer server
Primitive	PDU	Primitive	Primitive
		OPEN	CONNECT
SUBMIT	user MPDU	CLOSE	RELEASE
DELIVER		TRANSFER	DATA
PROBE	service MPDU	TURN-PLEASE	TOKEN-PLEASE
NOTIFY		TURN-GIVE	TOKEN-GIVE
		EXCEPTION	ACTIVITY-START
			ACTIVITY-INTER-RUPT
			ACTIVITY-RESUME
			ACTIVITY-END
			ACTIVITY-DISCARD
			SYNCHRONIZE-MINOR
			U-EXCEPTION-REPORT
			P-EXCEPTION-REPORT
			U-ABORT
			P-ABORT

of TRANSFER primitives. The message dispatcher may further use TURN-PLEASE and TURN-GIVE primitives to manage the turn for sending MPDUs in case the available association(s) is (are) two-way-alternate. It receives an EXCEPTION indication primitive carrying a previously submitted MPDU when the transfer of that MPDU could not be performed in the specified *transfer time* (a parameter of the TRANSFER request). After receipt of an EXCEPTION indication, rerouting the associated message may be attempted, or a service MPDU with a negative delivery report is generated.

The RTS uses the OSI connection-oriented presentation service, and through this the session service [11], to reliably transfer the user data specified in TRANSFER requests. A user data parameter is called here an APDU (application PDU); this is not an explicitly defined PDU. No PDUs are defined for the RTS.

Each APDU transfer constitutes a single *session activity*. After the start of a session activity, the APDU can be transferred in one or more presentation SDUs, each one submitted through a DATA request. Multiple DATA requests per APDU can only be used when *checkpointing* was agreed during connection setup. An APDU is then sent in parts, where each part is separated from the other through the insertion of a checkpoint. All checkpoints must be confirmed by the recipient RTS entity; the maximum number of unacknowledged checkpoints which may be outstanding during a session activity is indicated by the *window size* negotiated at connection establishment time. This is shown in *Fig. 4*. In case problems occur during the transfer of an APDU, which can be locally detected or signalled through U/P-EXCEPTION-REPORT or U/P-ABORT primitives, the sending RTS entity will attempt to *recover* the transfer with several possible actions, starting from the last confirmed checkpoint. We will not elaborate on this (note that several corrections and additional explanations w.r.t. RTS, especially covering recovery, are described in [6]). If the transfer cannot be completed within the allocated transfer time, the activity is normally discarded (ACTIVITY-DISCARD) and an EXCEPTION indication to the message dispatcher is generated by the sending RTS entity.

Analysis: It is typical that the definition of the message transfer protocol (that is, P1) does not mention the message transfer primitives which we characterized as being local. This results in an inconsistency between the protocol and service definition. We can make the following further remarks:

- the content of a message, i.e. user data, is not always transferred *transparently* by the message transfer protocol. For example, the message dispatcher may perform information type conversion of the user-provided content of a message. The conversion is not restricted to changing the representation of the user data, but may also include the translation to another data type.
- the transfer time parameter in a TRANSFER request primitive has only *local significance* and therefore does not have to be represented. The transfer time is commonly agreed by the mes-

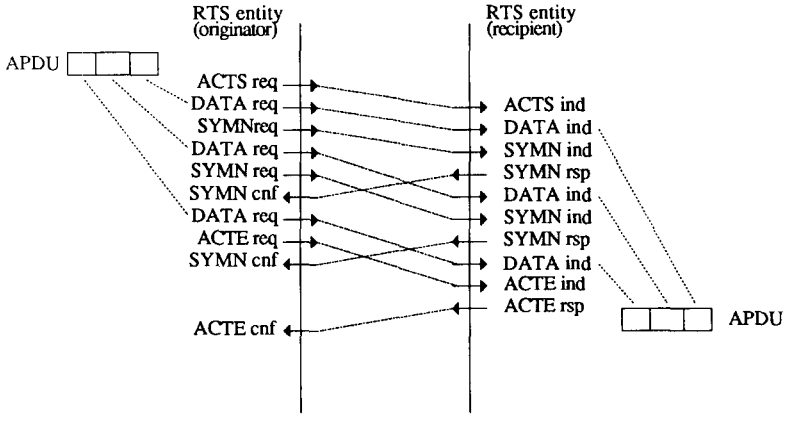

Fig. 4. Use of the presentation/session service for the transfer of an APDU in case checkpointing is used (here in 3 parts; checkpoint size is greater than zero and window size is at least two). (ACTS = ACTIVITY-START, ACTE = ACTIVITY-END, SYMN = SYNCHRONIZE-MINOR).

sage dispatcher and the local RTS entity at the sending side but is not visible at the receiving side.
- RTS defines a particular structure of the user data parameter of the presentation/session CONNECT primitives for transferring RTS-specific information, such as checkpoint size and window size. Since the OSI presentation service does not refer to this information, it seems that in this way an implicit RTS connect (-acknowledge) PDU is defined.
- the *correlation* between OPEN and CLOSE primitives is not described. CLOSE primitives carry no parameters: how then does the association manager indicate that it wants to delete an association with a particular MTAE? It is also not clear how the TRANSFER/TURN/EXCEPTION primitives are correlated with an association.

4.1.3 Message Transfer Interface

The possible distribution of a message transfer interface is represented in the X.400 recommendations as shown in *Fig. 5*. Two "concatenated" protocols, viz. the message transfer protocol (P1) and the submission and delivery protocol (P3), are used to provide the message transfer service. It should be noted that the P3 protocol is said to define the communication between an SDE and an MTAE, and not between two SDEs. The SDE functionality is thus "hidden" in such a MTAE. Another modeling of a distributed message transfer interface, consistent with the discussion in Section 4.1.1, is shown in *Fig. 6*.

The submission and delivery protocol is defined with the help of a general framework for interactive protocol definitions, referred to as *remote operations*. This framework defines four principal PDU data types, called OPDUs (for operation PDUs): Invoke, ReturnResult, ReturnError, and Reject. An Invoke OPDU specifies an operation; an entity sending an Invoke OPDU is said to

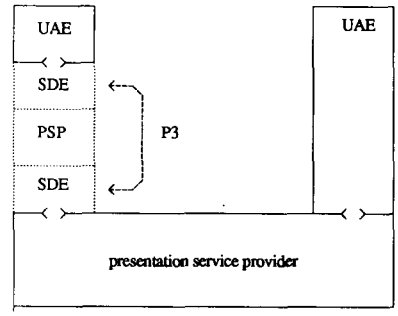

Fig. 6. Another view on "submission and delivery" (PSP: presentation service provider).

invoke a remote operation which must be performed by the recipient entity. Depending on the outcome of the operation, the recipient may return a:
- ReturnResult, reporting on the result of the operation when it was successful; or
- ReturnError, reporting on the error which occurred during the performance of the operation.

A Reject is sent on receipt of any of the Invoke, ReturnResult or ReturnError OPDUs when the OPDU was malformed and could not be processed for this reason.

For any specific protocol which makes use of the remote operations definition, hence also for the submission and delivery protocol, *particular operations* (and related results and errors) have to defined which are fit for that protocol. The submission and delivery protocol defines for all primitives listed in *Table 1*, except for the (UAL/MTL)LOGON and LOGOFF primitives, the associated operations. The so defined message transfer "interface" PDUs are transferred as user data on TRANSFER primitives of the RTS service, as described in Section 4.1.2. The (UAL/MTL)LOGON and LOGOFF primitives are directly mapped onto the RTS OPEN and CLOSE primitives.

Analysis: When we decompose an abstract in-

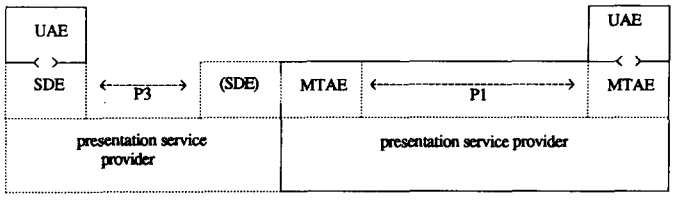

Fig. 5. "Submission and delivery" as modeled in MHS.

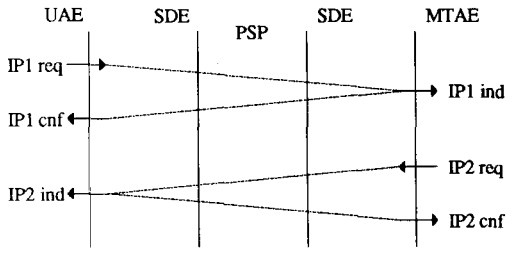

Fig. 7. Time sequences at a distributed message transfer interface of two (arbitrary) UAE-MTAE interactions, each one replacing a single service interaction (a request, IP1, and indication, IP2, respectively). (Internal mappings are not shown.)

terface (or SAP, represented by a vertical line in the time sequence diagrams of *Fig. 3* and *4*) of a service, we also have to decompose the service primitives which occur at that interface. *Fig. 7* shows a time diagram for such a decomposition, based on *Fig. 6*, and illustrates how a single service primitive can be represented as a "provider-confirmed" sequence of "interface" primitives. *Fig. 8* shows the specific case of submitting a message. On basis of these Figures we can conclude that:

- since the submission and delivery protocol defines the communication between an SDE and a MTAE (with embedded SDE), and not between two SDEs, the decomposition, or refinement, of the abstract message transfer interface is not very clear. It is for this reason, for example, that the (SUBMIT req) ind, shown in *Fig. 8*, is not explicitly specified, while the (SUBMIT req) req and the (SUBMIT req) cnf are. Similar omissions can be observed for the other interface elements. For the DELIVER and NOTIFY interface elements even two interface primitives, viz. the request and confirmation, are not described. The latter omission has important consequences as explained below.
- the DELIVER and NOTIFY interactions are not correctly described. Probably because there are no request and confirmation interface primitives specified, also the ReturnResult PDUs for the deliver and submit operations are not defined. Hence, in this case the acknowledgement of an operation is not only hidden at the invoker side, but completely omitted. This is in contradiction with *Fig. 7*.

The submission and delivery protocol relates to two sets of interactions. One is the set of interactions which are part of the service interactions described in the message transfer service, viz. SUBMIT, DELIVER and NOTIFY. The other concerns local activities, i.e. activities which involve no remote interactions (from the point of view of a message transfer service user) but only interaction between an UAE and its MTAE. This leads to the following comment:

- the submission and delivery protocol defines two sets of interactions which support different applications. These sets of interactions can be independently defined.

4.2 Interpersonal Messaging User Agent Layer (X.420)

Two PDU types are defined at this level: the intermessaging UAPDU and the status report UAPDU (UA for user agent). An intermessaging UAPDU consists of a heading and a body. The body contains one or more body parts, which can be looked

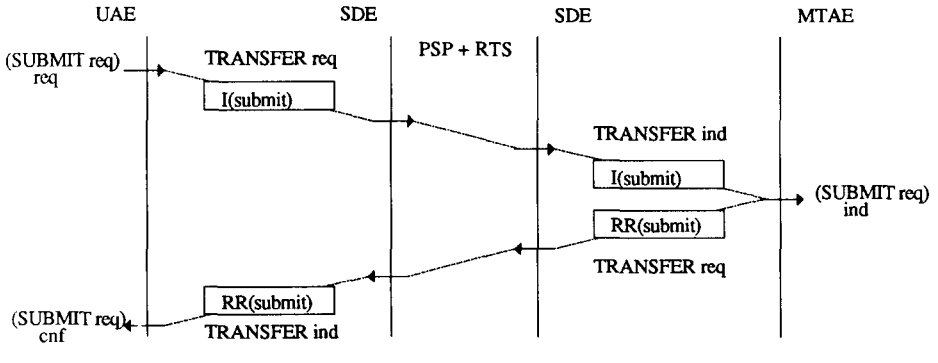

Fig. 8. Time sequence diagram for successfully submitting a message across a distributed message transfer interface (SUBMIT req, between brackets, indicates the original service primitive; I = invoke, RR = ReturnResult).

at as independent (sub)messages, with always an indication of the *body part type* (telex, teletex, voice, etc.). The heading always contains a *message identifier*, and optionally other interpersonnal messaging PCI. A status report UAPDU is used as an acknowledgement of the receipt or non-receipt of an intermessaging UAPDU; it therefore always carries the message identifier of the message to which it refers. Both PDUs are transferred by means of the SUBMIT/DELIVER service elements of the message transfer service. The interpersonal messaging protocol (P2) can provide the same facilities as listed in Section 2.2, on basis of the message transfer service, and some other facilities, including the following (a recipient interpersonal messaging service user is here shortly termed recipient):

- sending a message to one or more blind copy recipients, i.e. recipients which are not disclosed to the primary and secondary ("normal" copy) recipients specified in the request;
- notification of receipt or non-receipt (non-receipt means: receipt by the remote UAE, but not delivered to the intended recipient) of a message;
- delivery of messages which were auto-forwarded by the intermessaging protocol;
- conveyance of information as optional intermessaging UAPDU heading parameters, some of them on a per-message basis (the same information applies to all recipients in case of multi-recipient delivery), others on a per-recipient basis;
- transfer of a message consisting of several parts of possibly different types.

In addition, other, *management-like* functions are performed by UAEs which do not require the exchange of either of the above UAPDUs. Some of these functions concern the local access to the message transfer service and are not directly controlled by the interpersonal messaging service users. These functions are based on the use of the (UAL/MTL)LOGON, LOGOFF, REGISTER, (UAL/MTL)CHANGE-PASWORD, and (MTL)CONTROL. The other functions can be controlled by the interpersonal messaging service users; they are based on the use of the CANCEL, PROBE, and (UAL)CONTROL primitives.

Analysis: The following comments can be made:

- the interpersonal messaging *service* is poorly described. The service is not modeled by means of interrelated service primitives. Instead, the various service elements are outlined by indicating the effect of exchanging UAPDUs and the direct use of message transfer service (interface) primitives. The information which is exchanged in service interactions is not explicitly described, but must be derived from the UAPDU definitions or the message transfer service primitive definitions.
- the interpersonal messaging protocol describes the UAEs' engagement in both local and global activities. The same comments apply here as in Section 4.1.
- notification of successful delivery, provided by the message transfer service, is passed to the originating user of the interpersonal messaging service. This does not seem a very effective use of this service, as it only indicates a *probable* delivery to the peer user. Successful delivery can only be acknowledged by the receipt notification service element.
- some UAPDU heading parameters are not used by the interpersonal messaging protocol but have only relevance for the interpersonal messaging service users. This is the case with the optimal parameters which, if used, must be conveyed on a per-message basis: no interference of the interpersonal messaging protocol w.r.t. this information is required. It can therefore probably better be specified as a body part with an appropriate body part type.
- summarizing, it appears that the P2 protocol adds little value to the message transfer service. A part of the defined UAE operation concerns local management and does not require the cooperation with a peer entity; hence, such operation should not be described as part of the P2 protocol. Other definitions accrue from the need to distinguish between several user-relevant parameters, whose semantics must be correctly transferred (some of them only to a subset of the specified recipients) together with the actual message. Instead of mapping these parameters directly onto UAPDU parameters, a better design option seems to combine them in (recipient-bound) user data parameters with defined abstract syntaxes. In that case, the presentation service enables the correct interpretation of such data by the recipient peer user, while the data structure is not visible in the protocols supporting the users' interaction.

5. Message Handling within OSI

This section is concerned with the integration of MHS in OSI, where the message transfer service and protocol together constitute another *application service element* [2], based on the presentation service. Our aim is to investigate whether the presentation service is well utilized, and whether functions of the presentation service provider are not duplicated. In this context the RTS functionality is most suspicious; we will therefore concentrate on this functional part of MHS.

The OSI transport service provides a *reliable* and cost-optimized data transport capability. Depending on the quality of service requested by an initiating transport service user and the reliability of the underlying network, a suitable protocol *class* is negotiated between two transport entities (or the transport connection is refused).

The recovery procedures of RTS enhance the reliability provided by the transport service by enabling survival of protocol malfunctioning and connection losses (reported by EXCEPTION and ABORT primitives, respectively). They also duplicate part of the transport protocol functionality, since RTS is based on the assumption that only classes 0 and 1 can be negotiated by the transport protocol. In an OSI environment, only recovery of exceptional cases (network partitions, application crashes) should be left to an application protocol, whereas "normal" recovery can be delegated to the transport service provider.

The OSI session service enriches the transport service with the capability of exchanging data without imposing *length restrictions* and of *structuring* the communication (dialogue) between the users of the service.

Hence, checkpointing appears to be a redundant RTS functionality. The session protocol performs segmenting and reassembly to offer transfer of data of any length (recovery of data segments is performed by the transport service provider). Without checkpointing, and with the introduction of an RTS data- acknowledge PDU to obtain certainty about the acceptance of a data unit, selection of the activity management functional unit is not required any more. This might be advantageous for some implementations, given the fact that none of the current OSI application protocols makes use of activity services. Also the minor synchronize functional unit is not required in that case.

The OSI presentation service provides independence from the local *data representation* (encoding) in different systems involved in a communication.

RTS makes minimal use of the presentation service. On the other hand, considerable efforts were made by ISO to allow the conveyance of X.400 data by the presentation protocol. The reason for this is that X.409, which is the notation used for the definition of the X.400 PDUs, slightly diverges from the abstract syntax notation used by ISO. A universal treatment of data should be made possible in the presentation layer. The information type conversion function of the message transfer protocol also gives rise to some criticism. From an OSI point of view, representation of user data should be a concern of the presentation layer, and conversion from one to another datatype should be considered as an information processing task pertinent to a level above that which provides transparent transfer of the associated data, that is, the message transfer layer.

The entities which make up the OSI application layer are subdivided into entity parts, called application service elements (ASE). Corresponding ASES communicate according to a user-defined or standardized application protocol, where the latter may be either *application-specific* or *common* to most applications.

When MHS is to form a separate ASE in the application layer structure, it must also allow correct interworking in the presence of other ASES. Interworking of "composite" application entities is still under study in ISO TC97/SC21. As a final remark, it can be noted that the use of naming directories is currently described as an integral part of MHS. ISO defines separate service elements which allow common access to such directories. If this work is completed, other ASES will probably use the offered capability and include appropriate references to the relevant directory services.

6. Conclusions

Analysis of the X.400 recommendations gave rise to various points of criticism. Since the analysis was performed from two perspectives, two categories can be distinguished:

1. Misinterpretations of the architectural concepts

of service, protocol and interface. Among others, the following points are raised:

- Local and remote interactivities are mixed in the message transfer service definition.
- The service primitives used in describing the remote interactivities are not defined at the highest possible level of abstraction.
- P1 (message transfer protocol) is "concatenated" with P3 (submission and delivery protocol). In fact, P3 is a protocol which defines the interactions at the abstract interface between a UAE and a MTAE. Some of these interactions are a decomposition of the message transfer service primitives, which in turn define part of a remote interactivity. Others have no relation with message transfer service primitives since they have no corresponding remote effects.
- The decomposition of message transfer service primitives described by P3 is incomplete.
- The interpersonal messaging service is poorly defined. The protocol functionality which is added by P2 (interpersonal messaging protocol) is minimal.
- Transparent transfer of user data is not always performed by P1 and P2, contrary to what is claimed by the corresponding service or what could be expected from basic structuring principles.

It should be noted that these misinterpretations do not necessarily lead to wrong implementations. However, they blur the architecture and consequently impair the advantages of good structuring. For example, modelling errors may unnecessarily restrict implementations and may hamper correctness proofs; furthermore, they may lead to more complex implementations which are more difficult to test and to maintain.

2. Overdesign of the message transfer protocol as a consequence of disregarding lower layer functionality. In particular, the following observations are made:

- The RTS recovery procedures can be simplified given the service offered by the transport service provider.
- The RTS checkpointing function is redundant since the session service offers normal data transfer without length restrictions. The activity management and minor synchronize functional units are then no longer required for support of message handling.
- The existence of X.400 is visible in the presentation PDU definitions. Although suitable transfer syntaxes must be registrated for X.400 support, handling X.400 user data should not be different from any other user data.

Again, redundancy does not lead to wrong implementations. In this case, the architecture becomes unnecessary complex. It leads to implementation overhead and hence results in excess costs for subscribers to the service. For this reason it can better be avoided.

In addition to this basic criticism, a number of smaller defect have been discovered which were not discussed here. As has been shown in [3], such defects, including ambiguities, points of incompleteness and inconsistencies, can easily be discovered by using a formal description technique in defining the services and protocols. These techniques have the additional advantage of enlightening architectural aspects which remain vague in most informal texts.

It may be clear from the above that the positioning of MHS within the OSI reference model is problematic, in particular because OSI services and protocols are (should be) consistent with the concepts of service, etc. (which is not always the case, see e.g. [1]). In the light of the important application areas of message handling, the necessary adaptions should be agreed as soon as possible.

References

[1] I. Ajubi, M. v. Sinderen: "Design of a CCR Protocol Using a Formal Description Technique," submitted for inclusion in EUTECO '88 post-conference proceedings (North-Holland 1988).

[2] ISO: "Application Layer Structure," DP 9545, TC97/SC21 N1743 Revised, Oct. 1987.

[3] E. Dorregeest: "Analysis and Formal Specification of Electronic Mail," M. Thesis Report No. INF-87-3, Twente Univ., Enschede, The Netherlands, March 1987.

[4] T. Kalin (ed.): *Proc. of the European Telematic Conference (EUTECO), Workshop 1: Message Handling*, Varese, Italy, October 3–6, 1983 (North-Holland, 1983) 125–263, 631–640.

[5] W. Horak: "Office Document Architecture and Office Document Interchange Formats: Current Status of International Standardization," *IEEE Computer*, Oct. 1985, 50–60.

[6] CCITT: X.400-Series Implementor's Guide (Version 3)," COM VII-66-E (also: ISO/TC97/SC21 N1246), April 1986.

[7] CCITT: "Message Handling Systems," Recommendations X.400 ff., Red Book, Vol. 8, Fascicle 8.7, 1984.

[8] ISO: "Information Processing Systems – Text Communications – Functional Description of MOTIS," DIS 8505, TC97/SC18 N604, Feb. 1986.

[9] ISO: "Information Processing – Open Systems Interconnection – Basic Reference Model", IS7498, TC97, 1984.

[10] ISO: "Information Processing Systems – Open Systems Interconnection – Service Conventions," ISO TR8509, 1987.

[11] ISO: "Information Processing Systems – Open Systems Interconnection – Basic Connection Oriented Session Service Definition," ISO8326, 1987.

[12] C.A. Vissers, L. Logrippo: "The importance of the Service Concept in the Design of Data Communication Protocols," *IFIP WG6.1, 5th Int. Workshop on Prot. Spec., Ver. and Testing*, Toulouse-Moissac, France, June 10–13, 1985 (North-Holland, 1986) 3–17.