Proceedings of IFIP TC6 Workshop on Intelligent Networks

DATABASE ACCESS IN INTELLIGENT NETWORKS

Kimmo E. E. Raatikainen
University of Helsinki, Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23), FIN-00014 University of Helsinki, Finland

Abstract

Future telecommunication services will extensively exploit database technology. Information needed in operations and management of telecommunication services will be organized as a logical entity. The world-wide nature of telecommunication dictates that the logical entity can only be realized through interoperability of autonomous databases that may have different internal characteristics. We review the requirements for database access in the IN architecture. Our premises are derived from the IN long-term architecture framework as specified in ITU-T Recommendation Q.1201.

1. INTRODUCTION

Databases will have a dominant role in telecommunication services based on the Intelligent Network architecture. The databases hold the information needed in operations and management of telecommunication services. The performance, reliability, and availability requirements of data access operations are hard. Thousands of retrievals must be executed in a second. Allowed down-time is only a few seconds in a year.

The globalization of services implies that databases managed by different operators must interoperate. This requires that the systems are open, use standardized protocols, are protected against misuse and intruders, and have a common logical view of information.

In this paper we review ITU-T Recommendations in Q.1200 (Intelligent Network), X.500 (Directory), and X.700 (OSI Management) Series. Our objective is to derive the requirements for database access in the IN architecture. The paper is organized as follows. In Section 2 we examine the IN long-term architecture framework. In Section 3 we examine the current Recommendations in Q.1200 Series. We also discuss the next Refinements based on the X.500 Directory. The X.500 Series of Recommendations are summarized in Section 4 and X.700 Series in Section 5.

2. PERSPECTIVE

The *Intelligent Network long-term architecture framework* is the structure that allows the integration of technologies developed in other standards activities into the *IN architecture*. Of these activities the most important are *Open Distributed Processing* (ODP) and *Telecommunications Management Network* (TMN). The IN framework will provide an open architecture that is achieved through the integration of computing information and telephony technologies. The architecture will be enhanced by emerging technologies including broadband capabilities, distributed processing, Open Systems Interconnection, object-oriented modelling, information technology, cooperative processing, distribution control, management of services and networks, verification/validation, and artificial intelligent.

It is intended that the *IN conceptual model* as described in Recommendation Q.1201 *Principles of Intelligent Network Architecture* remains consistent throughout the evolution of the IN architecture. In Q.1201 there is a list of attributes that intelligent networks will have. The list includes integrated services, integrated/shareable control, programmability, adaptability facilitated by modularized hardware and software, interoperability of networks and systems, and an OSI-aligned protocol architecture for all interfaces that facilitate communication between entities. For the database access the last three items are the the most important ones.

The need for interoperability calls for openness, particularly for open distributed processing. From the computational viewpoint[1], the services and the computing platforms can be represented as an aggregation of collections of objects. This object-oriented modelling is currently considered as the easiest way to provide a framework for mechanisms that implement the necessary transparencies including access, concurrency, failure, location, and replication transparency.

The *location transparency* hides the location of an object. The *access transparency* gives a similar access to the methods of local and remote objects. The *concurrency transparency* hides the existence of other concurrent users of an object. *Replication transparency* hides the effects of having multiple copies of an object while *failure transparency* provides fault tolerance. In brief, the transparencies enables greater freedom and independence in the design of applications and services.

It is intended that the IN long-term architecture takes advantage of new and emerging technologies as appropriate. Several of these technologies were mentioned at the beginning of this section. Among them were distributed processing, Open Systems Interconnection, object-oriented modelling, and management of services and networks.

Distributed processing is the mechanism for maintaining an environment which is composed of a variety of applications, protocols, and platforms. Important issues related to distributed processing include scalability, portability, and performance. Scalability and portability are the ways to achieve seamless and cost-efficient evolution of the computing platform. Data communication services that function as data transport services are an essential area in distributed processing, for its performance and fault tolerance. The concepts of database management must also be applied in a distributed environment. Database services are need to access and manage structured elements through the management of information processing and the database infrastructure.

The OSI Reference Model permits interworking between different systems. OSI is concerned not only with the transfer of information between systems, but also with their capability to interwork in order to achieve a common distributed task. For the database access, the higher protocol layers, particularly the application layer together with the

[1] Viewpoints are pragmatic tools defined in the ISO *Reference Model of Open Distributed Processing* (RM-ODP). Each viewpoint — enterprise, information, computational, engineering, and technology — leads to a representation of the system with emphasis on a specific concern. Within IN the notion of viewpoints has been adopted in the concepts of planes in the IN Conceptual Model. It is important to realize that multiple viewpoints must be considered to ensure portability. A recent introduction to RM-ODP can be found in Raymond [1993].

OSI-applications — such as Directory (X.500), Security (X.800), and Transaction Processing (X.860) — is of fundamental importance. In addition, we cannot forget the management aspects of the OSI architecture (X.700).

Object-oriented modelling is pointed out by the following quotation in Q.1201: *"The use of object modelling could satisfy the modelling needs of IN long-term architecture by the use of abstract object modelling concept."* Our claim is even stronger than that. We are convinced that today object modelling is the easiest way to satisfy those modelling needs.

The object-oriented methodology allows strict modular system specifications. The strong encapsulation supported by object-oriented design is a prerequisite for evolutionary changes within a specification so that the system can be regarded as an open system. The encapsulation hides all changes in the implementation of objects as long as the new interfaces to the methods remain compatible with the old ones.

In order to ensure that large distributed systems are tractable, the systems must be designed in a way that minimizes the interdependencies between the components in the system. Object-oriented modelling techniques are particularly well suited to these purposes because they provide the benefits of abstraction, encapsulation, and modularity. Abstraction is a tool that simplifies the description of systems through describing only those characteristics that are meaningful on the current description level. Encapsulation is a technique for only exposing the observable behaviour of an object's services and providing clients with information how to invoke these services. On the other hand, encapsulation hides the details of the object's implementation. Modularity is achieved because a system can be specified as an aggregation of collections of objects. The fact that subsystems can be treated as independent objects greatly simplifies system descriptions.

Object modelling promotes modularity by enabling the structuring of specifications into smaller parts. By constraining all interactions between objects to take place at well defined interfaces, the interdependence of objects is minimized. By isolating and explicitly describing all interfaces between objects, the dynamic and evolutionary nature of distributed systems can be more easily modelled.

For the database access, the ongoing standardization in the area of object database management is seminal. A consortium has developed one standard called ODMG-93 Standard [Cattell, 1994]. Other standardization activities include the Object Data Model by the Object Management Group (OMG); see e.g. OMG [1992] and Soley [1992]. Unfortunately, the field of object-oriented database management is not yet mature. However, the group developing the RM-ODP is strongly committed to be compatible with the OMG specifications.

Management of services and network requires that the IN architecture and TMN concepts are integrated. In IN, managed object are very diverse. In order to realize the management of all the different kinds of managed objects in practice, the Management Information Base (MIB) must have an efficient implementation. This challenges the database architecture by implying a twofold functionality. Firstly, the database and its elements are managed objects. Secondly, the database could be the tool to implement the MIB.

To summarize, we have reviewed and briefly commented the Recommendation Q.1201. Our goal was to introduce the premises and requirements for a database architecture that can be deployed as the Service Data Function. The key requirements are 1) an OSI-aligned protocol architecture for all interfaces, 2) compatibility with the OSI Reference Model of Open Distributed Processing (RM-ODP), and 3) adapting ITU-T Recommendations for Telecommunications Management Network (TMN). In particular, the two last requirements call for an object-oriented approach.

3. GUIDELINES FOR IMPLEMENTING DATABASE SERVICES IN CURRENT Q.1200 SERIES RECOMMENDATIONS

3.1 FUNCTIONALITY

The *distributed functional plane* (DFP) in the INCM consists of *functional entities* (FEs) and of the relationships between the FEs. Database services are provided by the FE called *Service Data Function* (SDF). Recommendation Q.1204 *Intelligent Network Distributed Functional Plane Architecture* specifies that the SDF contains customer and network data for real time access by the FE called *Service Control Function* (SCF) in the execution of an IN provided service. The SDF interfaces and interacts with SCFs and other SDFs. The SDF is managed, updated, and otherwise administered by an FE called *Service Management Function* (SMF).

It should be noted that the SDF contains data which are directly related to the provision of IN provided services. Therefore, the SDF does not necessarily encompass data provided by a third party but may provide access to these data. Examples of service data processing functions accessible to the SCF from the SDF include functions to access service information (e.g. subscription data parameters) and to update service information (e.g. sum of charging).

Capabilities in the Capability Set 1 (CS-1) are intended to support services[2] and service features[3] that fall into the category of "single ended", "single point of control" services. Such services are referred to as Type A services. All other services are placed in a category called Type B. Due to operational, implementation, and control complexity CS-1 standards do not encompass Type B services.

Recommendation Q.1211 *Introduction to Intelligent Network Capability Set 1* augments the functional specification of the SDF through requiring that SDF provides consistency checks on data. This implies that SDF should provide the transaction processing capabilities. Q.1211 also specifies that SDF hides the real data implementation and provides a logical data view to the SCF.

The service management aspects primarily address the network operator's interaction with the SCF, SDF, an FE called *Service Switching Function* (SSF), and an FE called *Specialized Resource Function* (SRF). Since this interaction normally takes place outside the context of a particular call or service invocation, the treatment of management aspects is only cursory in the CS-1 Recommendations. However, Q.1211 requires that CS-1 must neither exclude nor constrain the capability of service customers to interact directly

[2]A service is a stand-alone commercial offering, characterized by one or more service features, and can be optionally enchanced by other service features.

[3]A service feature is a specific aspect of a service that can also be used in conjunction with other services/service features as part of a commercial offering. A service feature is either a core part of a service or an optional part offered as an enhancement to a service.

with customer-specific service management information. A personal service profile is an example of such information. Moreover, the following two points may be relevant to the CS-1 timeframe. Firstly, the SMF, an FE called *Service Creation Environment Function* (SCEF), an FE called *Service Management Access Function* (SMAF) may be used to add, change, or delete CS-1 based service related information or resources in the SSF, SCF, SDF, and SRF. Such changes should not interface with CS-1 based service invocations or calls that are already in progress. Secondly, the network operator may, at its discretion, give the service customer the ability to add, change, or delete appropriate customer-specific information. The mechanisms and safeguards that are put into place by the network operator for this interaction may take advantage of CS-1 functions and capabilities.

3.2 RELATIONSHIPS AND INFORMATION FLOWS

In the DFP each interaction between a communication pair of FEs is termed an *Information Flow* (IF). The relationship between any communicating pair of FEs is defined by a set of information flows. If a communicating pair of FEs is located in physically separate entities, the relationship between them defines the information transfer requirements for a protocol between the physical entities. In order for one FE to invoke the capabilities provided by another FE, a relationship must be established between the two FEs concerned. This implies that the SDF is a server and that SCFs and SMFs are its clients. It should be noted that a relationship can only be established by the client FE.

The relationships recognized in Q.1204 include SCF-SDF, SDF-SDF, and SMF-SDF. Furthermore, Q.1211 identifies 18 distinct functional relationships, of which five are related to network interworking. Q.1211 identifies the relationships as reference points and assigns a unique one-letter identifier to each reference point. CS-1 divides the relationships into four control classes: 1) connection-control capabilities, 2) call-control (Non-IN service-control) capabilities, 3) IN service-control capabilities, and 4) management-related control capabilities. The difference between classes 2 and 3 is that the capabilities in class 3 involve the structured separations of the SSF from the SCF whereas the capabilities in class 2 do not involve that separation.

Figure 1 illustrates the relationships between SCFs, SDFs, and SMFs in two different networks. The functional relationship at a reference point may provide for one or more control classes. Each combinations of a functional relationship and a control class is referred to as a control relationship and is identified by an `<alpha>.<number>` string, where `<alpha>` identifies the functional relationship and `<number>` identifies the control class.

The physical aspects of the realization of each functional relationship do not imply a direct physical interface between the involved network functions. The CS-1 standardization defines the IN *application service elements* (ASEs) independent of the underlying protocol stack. However, it is recommended that the IN ASEs should be used with existing standardized protocol stacks. Control relationships involving the SDF are F.3, H.3, P.3, and Q.3, of which F.3 and P.3 are within the scope of CS-1. Q.1211 recommends SS No. 7/TCAP and DSS 1/Q.932 to be used for F.3. In addition, the final sentence in Clause 7.7 of Recommendation Q.1211 is noteworthy: *Wherever possible, the CS-1*

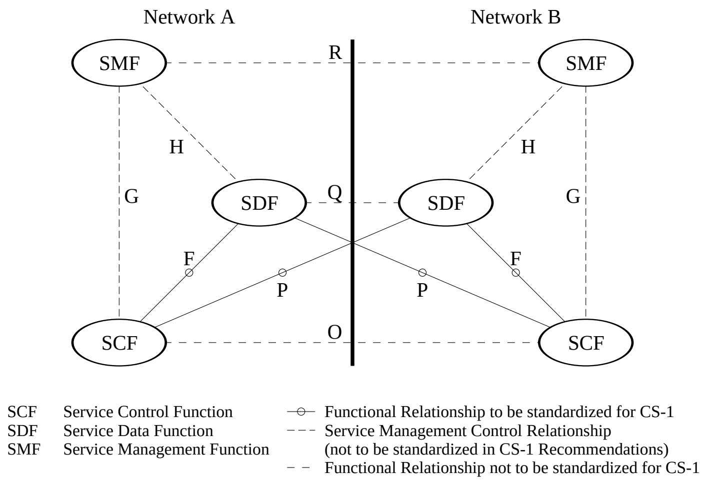

Figure 1: Functional Relationships and Reference Points for CS-1

Recommendations identify alternative interfaces (e.g. SCCP-GTT, X.500, or CMISE) for these domains.

The ASEs at different reference points should be defined separately within a common structure. This helps to develop a modular and flexible *IN application protocol* (INAP). The INAP in turn, facilitates flexible packaging of the functional elements in the DFP into a variety of different *physical elements* (PEs) in the *physical plane* of the INCM. The decomposition of standardized *service independent building blocks* (SIBs) in Recommendation Q.1213 *Global Functional Plane for Intelligent Network CS-1* have been the basis for determining the number, nature, and content of the ASEs. In addition to the Basic Call Processing SIB, Q.1213 specifies 14 SIBs. The execution of the following five SIBs need the support of SDF: Log Call Information (LCI), Screen, Service Data Management (SDM), Status Notification, and Translate.

The definition of INAP ASEs reflects the capabilities that can be differentially applied to the three separate classes of services. Class 1 includes the services that benefit from IN service control in call set-up and tear-down phases. Class 2 includes the services that require mid-call control. Class 3 includes the services that require topology manipulation. The focus in CS-1 has been on Class 1. CS-1 can only support a limited use of mid-call and topology manipulation capabilities.

Recommendation Q.1214 *Distributed Functional Plane for Intelligent Network CS-1* divides the information flows into nine categories. Six of these categories fall into Class 1. Category r4 contains the information flows (IFs) between the SCF and SDF. The IFs in category r4 are: Query, Query Result, SDF Response, Update Confirmation, and Update Data.

Query is the IF that is used by the SCF to retrieve an item of data held in the SDF. It has three *information elements* (IEs): Database ID, Requested Info type, and Information key. The reply to Query by the SDF is the Query Result IF which has the IE Requested Info.

Update Data IF when requested by an SCF will entail an atomic execution of the update. However, problems such as concurrent access to the data are not solved by the IFs. This IF has four IEs: Function type, Database ID, Updated Info, and Information key.

Update Confirmation IF is the response to the Update data IF with the IE Outcome. The SDF sends this IF to an SCF to provide the result of writing to a specified service data object. The IE Outcome describes the result of the request operation, either success or failure with a specific reason.

The SDF may issue the information flow **SDF Response** to the SCF as an interim response to the Query or Update Data IF. This IF is an indication that the request has been received but may take some time to execute. The IF will subsequently be followed by either a Query Result IF or an Update Confirmation IF.

The IE Database ID identifies the logical location of the database in which the requested information resides. It does not refer to a particular service, but rather to some specific data. The IE Requested Info type identifies the information whose value is requested. The structure of this IE is to be defined within each specific database.

Information key IE is used to locate the requested information fields. IEs in the initial DP IF[4] are all candidates for information key. The precise structure and possible values for the IEs will be service specific.

The information elements Function Type in Update data IF is used to indicate the action to be carried out on the particular data. Possibles values are replace, increment, and decrement. Update Info IE gives the new value of the data to be modified if the function type is replace. It gives the value by which the data is incremented (decremented) if the function type is increment (decrement).

3.3 SERVICE PROCESSING MODELS

A high level overview of a desirable IN service processing model is illustrated in Figure 2. The three main elements of this model are: 1) the basic call processes, 2) the "hooks" that allow the basic call processes to interact with IN service logic, and 3) IN service logic that can be "programmed" to implement new supplementary services. Q.1211 gives a more detailed description: SCF receives and decodes the query, and interprets it in the context of a CS-1 supported service. It formulates, encodes, and sends a standardized response to the SSF. The formulation of the response may involve complex service logic leading to a query to a separate SDF.

In Q.1214 we can find further refinements. An intelligent network has two realms related to call/service processing: 1) basic call processing and 2) service control. Service control resides in the SCF entity. It interacts with basic call processing via an SSF entity associated with the FE called *Call Control Function* (CCF).

[4]The information flow Initial DP is issued from an SSF to the SCF. It is generated when the SSF detects a trigger at any *detection point* (DP) in the *Basic Call State Model* (BCSM) in order to request instructions from the SCF. The IEs of Initial DP are: Call ID, Service key, Call gapping encountered, Dialled digits, Called party number, Calling line identity, Calling party category, SSF/SRF capabilities, SRF available, Misc call info, Terminal type, Service profile identifier, Location number, Calling party business group ID, Calling party sub address, Original called party ID.

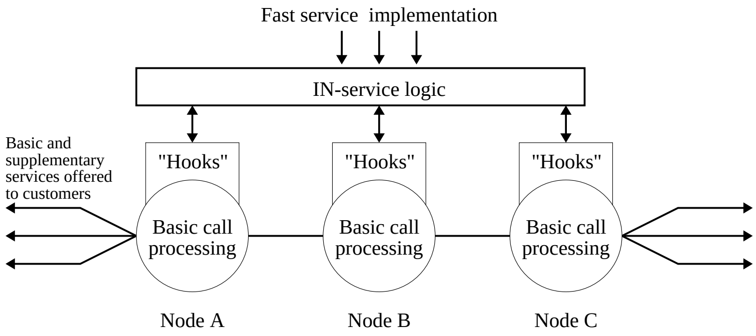

Figure 2: IN service processing model (Figure 16/Q.1201)

A relationship is established between the SCF and SDF at the request of the SCF when the SCF requires to receive or modify some data contained within the SDF. The relationship is terminated by the SDF. Information flows related to the SDF may be associated with some degree of processing that depends on the supported service. This processing is related to data manipulation but not to call processing. Only logical view of data is known to the SCF. The IFs do not imply any physical organization of data or how they are stored. In particular, the fact that data are replicated is not known to the SCF.

To realized the functionality needed in the SCF, Q.1214 provides an SCF model shown in Figure 3. It should be noted that the model is conceptual and that it is not intended to imply an actual implementation of the SCF. However, the model provides a useful framework for a client of database services.

Let us briefly examine the SCF model. The *Service Logic Execution Manager* (SLEM) is the functionality that handles and controls the total service logic execution action. The SLEM interacts with *SCF Data Access Manager* and *Functional Entity Access Manager* to support the execution of an *Service Logic Processing program Instance* (SLPI). In particular, the SLEM needs functionality to manage SLPI access to SCF and SDF data via the SCF Data Access Manager.

The SCF Data Access Manager provides the functionality needed to provide for the storage, management, and access of shared and persistent information in the SCF, that is the information persisting beyond the lifetime of a SLPI. The SCF Data Access Manager also provides the functionality needed to access remote information in SDFs. The SCF Data Access Manager interacts with the SLEM to provide these functionalities to SLPIs.

The SCF data is contained in the *Service Data Object Directory* and in the *IN Network-Wide Resource data*. The SDF Data Access Manager uses the Service Data Object Directory to locate service data objects in the network in a manner transparent to the SLEM and its SLPI. As such, the SLEM and its SLPIs have a global and uniform view of service data objects in the network.

The *Functional Entity Access Manager* (FEAM) provides the functionality needed by the SLEM to exchange information with other functional entities via messages. This message handling functionality should: 1) be transparent to SLPIs, 2) provide reliable mes-

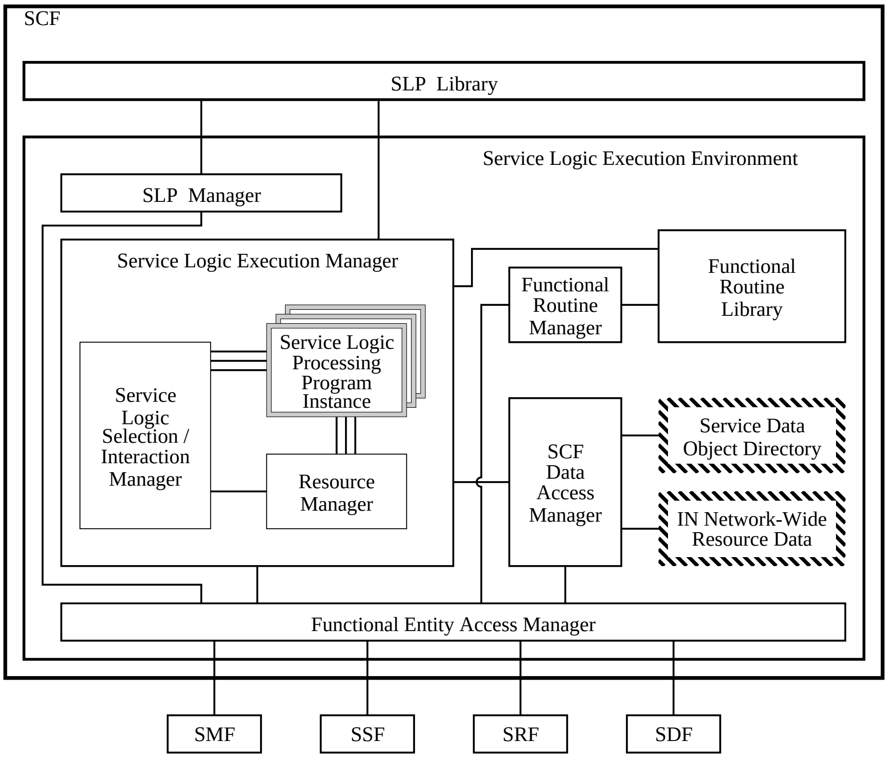

Figure 3: Service Control Function Model (Figure 4-19/Q.1214)

sage transfer, 3) ensure sequential message delivery, 4) allow message request/response pairs to be correlated, 5) allow multiple messages to be associated with each other, and 6) comply with OSI structures and principles.

In addition to the SCF Model, Q.1214 provides a conceptual model of the SDF, which is shown in Figure 4. The SDF contains and manages the data which are related to Service Logic Processing programs (SLPs). The data are accessed in the execution of the SLP instances. Therefore, data such as SLP selection data and SCF directory, which are accessed before the execution of an SLPI, are not included in the SDF handling data.

The functional entities in the SDF model are *SDF Data Manager*, *Exclusive Control Manager*, and *Functional Entity Access Manager*. The SDF Data Manager provides the functionality needed for storing, managing and accessing information in the SDF.

The Functional Entity Access Manager (FEAM) provides the functionality needed by the SDF Data Manager to exchange information with other functional entities (i.e. SCF, SDF, and SMF) via messages. This message handling functionality should: 1) provide reliable message transfer, 2) ensure sequential message delivery, 3) allow message request/response pairs to be correlated, 4) allow multiple messages to be associated with each other, and 5) comply with OSI structures and principles. The FEAM may access other SDFs, because the data distribution in the network can completely be transparent to the SCF. However, this point as well as the functional relationship with the SMF is outside the scope of CS-1.

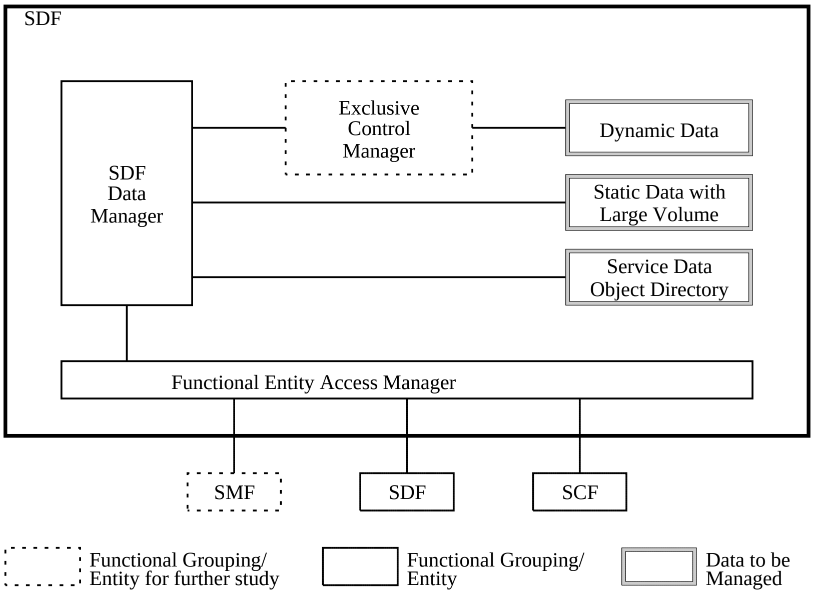

Figure 4: Service Data Function Model (Figure 4-20/Q.1214)

The Exclusive Control Manager provides the functionality needed to provide exclusive control, for example lock-unlock control, to ensure data integrity. However, this manager and its action method are identified as items for further study.

The data handled by the SDF is classified into types of "static" and "dynamic". The data that are "read-only" as far as SLPs are concerned are called static. The data that can be changed by SLPs are called dynamic. It should be noted that these definitions of "static" and "dynamic" are different from the definitions given in Recommendation Q.1290 *Vocabulary of Terms Used in the Definition of Intelligent Networks*.

The types of data in Q.1214 are further subdivided into six types. *Type 1 data* are dynamic data that are local to an SLPI, e.g. call instance data parameters like the dialled number. *Type 2 data* are static data that are feature-specific and are shared by SLPIs, e.g. subscription data parameters like day of week or time of day screening. *Type 3 data* are dynamic data that are feature-specific and are shared by SLPIs, e.g. sum of charging or a counter for a call number limiting service. *Type 4 data* are static data that belong to multiple service features and are shared by SLPIs, e.g. a subscriber's phone number list to connect. *Type 5 data* are dynamic data that belong to multiple service features and are shared by SLPIs, e.g. subscriber's location data used by a service such as UPT. *Type 6 data* are data in the Service Data Object Directory.

It is assumed that an SLP includes type 1 data. Besides the locally available service data objects, additional data is used to locate service data objects in other SDFs in the network. The additional data is used in a manner which is transparent to the SLEM and its SLPI in the SCF requesting the locally unavailable data.

Upon a data object retrieval request by the SCF, the SDF Data Manager will try to locate the data object locally. When the requested data object is not available, it will try to retrieve a reference to another SDF from the Service Data Object Directory. If

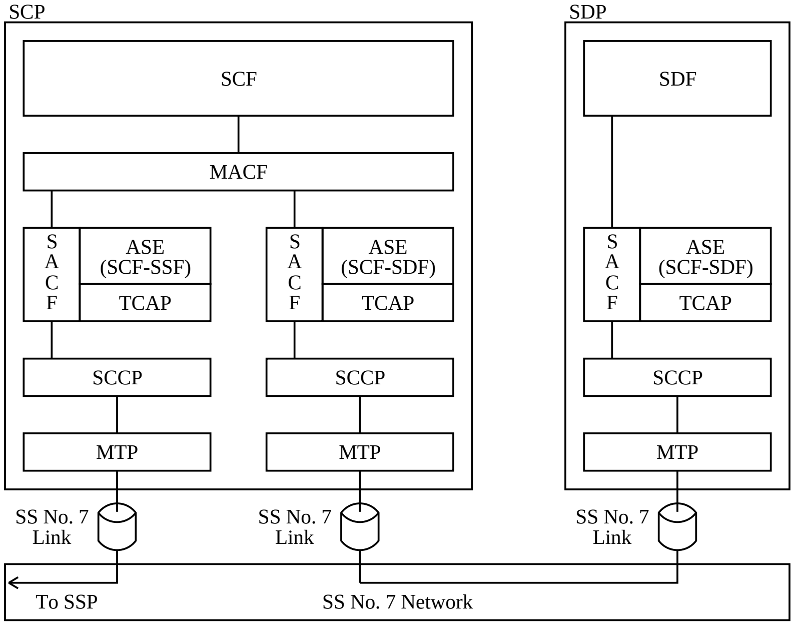

Figure 5: Physical Interface Between SCP and SDP (Figure 1/Q.1218)

the reference is available, the SDF will either refer this back to the requesting SCF, or try to retrieve the requested data directly from the referenced SDF. However, the latter mechanism is outside the scope of CS-1. If a reference is not available, the SDF Data Manager will return a failure to the requesting SCF.

3.4 INTELLIGENT NETWORK APPLICATION PROTOCOL

The intelligent network application protocol (INAP) that is required for support of CS-1 is defined in Recommendation Q.1218 *Interface Recommendation for Intelligent Network CS-1*. The definition of INAP can be split into three sections: 1) the definition of *Single/Multiple Association Control Function* (SACF/MACF) rules for the protocol, 2) the definition of the operations transferred between entities, and 3) the definition of the actions taken at each entity.

The INAP is a ROSE user protocol. The ROSE protocol is contained within the component sublayer of TCAP (Recommendations Q.771 to 775) and DSS 1 (Recommendation Q.932). The ROSE *Application Protocol Data Units* (APDUs) are conveyed in transaction sublayer messages in SS No. 7 and in the Q.931 REGISTER, FACILITY, and call control messages in DSS 1. The INAP (as a ROSE user) and the ROSE protocol have been specified using the Abstract Syntax Notation One (ASN.1). At present, the only standardized way to encode the resulting PDUs is the Basic Encoding Rules (BERs).

The INAP will support any mapping of functional entities (FEs) to physical entities (PEs). Therefore the protocol is defined assuming maximum distribution, that is one PE per FE. The physical interface between an SCF locating in a *Service Control Point* (SCP) and an SDF locating in a *Service Data Point* (SDP) is illustrated in Figure 5. The interface will be INAP using TCAP which, in turn, uses the services of the connectionless

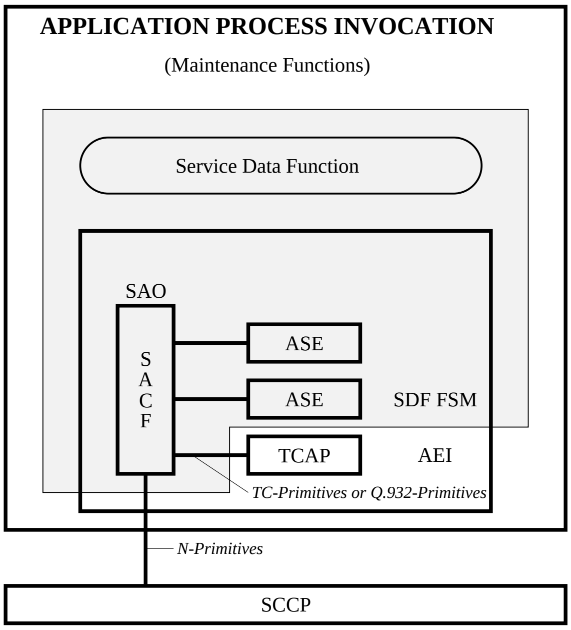

Figure 6: Functional Model of SDF Application Entity (Figure 41/Q.1218)

SCCP and MTP. The SDF is responsible for any interworking to other protocols to access other types of networks.

The INAP is the collection of all specifications in all ASEs. Each ASE supports one or more operations. Each operation is tied with the action of corresponding functional entity. The use of the application context negation mechanism as defines in the Q.770-Series (*Transaction capabilities application part*) allows the two communicating entities to identify exactly what their capabilities are and what the capabilities required on the interface should be. If the indication of a specific application context is not supported by a pair of communicating FEs, some mechanism to pre-arrange the context must be supported.

TCAP Application Context (AC) negotiation rules require that the proposed AC, if acceptable, is reflected in the first backwards message. If the AC is not acceptable, and the TC-User does not wish to continue the dialogue, it may provide an alternate AC to the initiator which can be used to start a new dialogue.

The *SDF Application Entity* (AE-SDF) includes TCAP and one or more ASEs called TC-users. The functional model of the AE-SDF is shown in Figure 6. The shaded area in the figure indicates the scope of Recommendation Q.1218. The ASEs interface to TCAP to communicate with the SCF. They also interface to maintenance functions. The interfaces use the TC-user ASE primitives specified in Recommendation Q.771 and N-Primitives specified in Recommendation Q.711. The operations of INAP are Query, UpdateData, and SdfResponse. They correspond to the information flows Query, Update Data, and SDF Response. Observe that the IFs Query Result and Update Confirmation are mapped on to operation Return Result from Query and UpdateData, respectively.

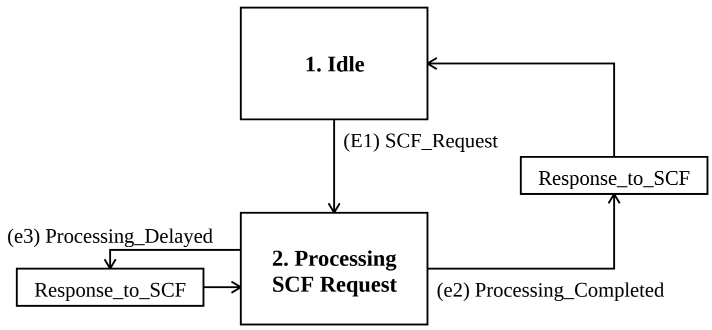

Figure 7: The SDF Finite State Model (Figure 42/Q.1218)

As far as CS-1 is concerned, the function of SDF is to synchronously respond to every request from the SCF. Therefore, the respective *Finite State Model* (FSM) shown in Figure 7 is trivial. In any state, if there is an error in a received operation, the maintenance functions are informed and the SDF FSM remains in the idle state. In addition, the error can be reported to the SCF using the appropriate component (see Recommendation Q.774). In any state, if the dialogue with the SCF is terminated, then the SDF FSM returns to idle state after ensuring that any resources allocated to the call have been deallocated.

The only event accepted in state "Idle" is `SCF_Request` caused by the reception of Query or UpdateData operation from the SCF. This event causes a transition to state "Processing SCF request".

The events accepted in state "Processing SCF request" are `Processing_Completed` and `Processing_Delayed`. The event `Processing_Completed` occurs, when the SCF request completes. This event causes the response to the Query or UpdateData operation to be sent to the SCF. In addition, the FSM transites into state "Idle". The event `Processing_Delayed` is an internal event caused by the recognition that the SCF request may be delayed. In order to inform the SCF about this situation, the SDF FSM sends the SdfResponse operation to the SCF.

It is important to keep in mind that for the network interworking purposes, the SDF may return to the SCF a reference to another physical entity containing the requested information.

3.5 IDENTIFIED PROBLEMS AND PROPOSED SOLUTIONS

The CS-1 specifications have an ASE called *User Data Manipulation ASE* that provides the necessary communication capabilities to access an SDF from an SCF. The ASE consists of two operations: `Query` and `UpdateData`. In brief, the User Data Manipulation ASE is very simple, obviously oversimplified. Attempts to use the operations included in the ASE to support advanced services such as Universal Personal Telecommunication (UPT), Universal Mobile Telecommunication System (UMTS), or Virtual Private Network (VPN) have, according to Chatras and Gallant [1994], shown several shortcomings.

The two main concerns are the service-dependent semantics of the parameters in the operations and the authentication. The service-dependent semantics of the parameters imply that the actions performed by the SDF are also service-dependent. This means that the SDF must be separately reimplemented or tailored for each service. Therefore, the current specification of User Data Manipulation ASE contradicts the principles of IN long-term architecture, since the SDF behaviour must be service-independent.

The second main concern is the authentication. The globalization of services implies that SDFs will be accessed by foreign SCFs. Therefore, the SDF must perform the authentication of users and provide the access control. In the current specification the only way to carry out authentication is to use ordinary attribute value comparison. Therefore, the SDF does not know when authentication takes place.

To overcome the problems in the SDF access ETSI and later ITU-T SG11 have proposed that a new SCF-SDF interface is needed. This interface, which will be standardized in the near future, is based on the *Directory Access Protocol* (DAP) specified in X.500 Series of Recommendations.

When we look for the reasons why the database access in CS-1 Recommendations came to a dead-end, the explanation given most often is that the SDF operations were defined quite lately, on the fly, without any deep analysis of their use [Chatras and Gallant, 1994]. However, we believe that the fundamental reason was the current commercial database products based on the relational data model. The relational data model, which is inherently flat, has simple operations. However, the information needed in IN-services is strongly structured, more like a hierarchical tree than a table. Therefore, the lack of an explicit information model for CS-1 Recommendations is the main source of problems leading to the dead-end.

4. X.500 DIRECTORY

The next refinement of the SCF-SDF interface will be based on a subset of the *Directory Access Protocol* (DAP). According to Chatras and Gallant [1994] the time constraint was a determinant factor for the choice of DAP in CS-1 Refinements. The DAP solution was preferred to CMIS (X.700) for the following two reasons: 1) unambiguous security and authentication model and 2) distribution and replication procedures can be easily introduced.

In this section we give a brief introduction to the X.500 Directory. We emphasize the features and properties that are significant for our theme. Comprehensive introductions to X.500 can be found, for example, in Sykas and Lyberopoulos [1991] or in Bumbulis et al. [1993].

4.1 INFORMATION MODEL

The Directory can be viewed as a world-wide logical entity that contains all pieces of relevant information. In order to keep the Directory manageable and fault-tolerant, the Directory is distributed and replicated.

The information contained in the Directory is stored in *entries* and is organized in an information tree called *Directory Information Tree* (DIT). Collectively, the entries in the Directory are referred to as the *Directory Information Base* (DIB). Each entry in the

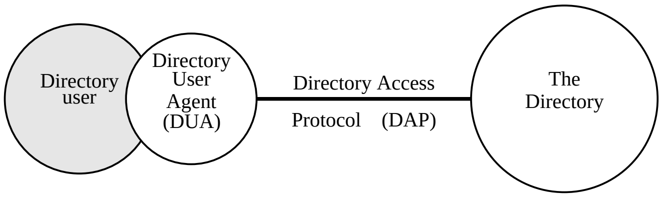

Figure 8: Directory Access Model

DIB holds information about a single object. These pieces of information are organized as a set of *attributes*. Each attribute holds a certain piece of information about a single facet of the object. An attribute has the *attribute type* that indicates how the associated *attribute value* is to be interpreted. An attribute may be multi-valued. Each attribute type has an associated *attribute syntax* and *matching rule*. The syntax describes how the information of this type is represented in requests and replies. The matching rule determines how values of this type are to be compared. Typically, an attribute type is specified as a *subtype* (refinement) of some more general type. A subset of each entry's attribute values are distinguished to form the *relative distinguished name* (RDN) which is the unique (with the respect to all siblings of a common parent) name of this entry.

The *class* of a Directory entry is determined through the type of object that it represents. Entries of a particular class must have certain *mandatory attributes*. They may also have other *optional attributes*. Object classes are organized into a hierarchy according to "is-a" relationship. The class hierarchy makes it easy to create new classes that are extensions of existing classes. A special entry class Alias is defined to refer to an object by more than one name. The aliases imply the actual structure of DIT is directed graph.

To summarize, the Directory provides an object-oriented information model. X.500 information model is quite flexible but not as general-purpose as the information model in ODMG-93 [Cattell 1994] and in OMG [OMG 1992]. The information model permits generic operations which are parametrized by the data carried in the operations. The model is a formal description of the data that is needed to support the services. In addition to the formal description, the information model of X.500 provides a formal organization of the data.

4.2 DIRECTORY ACCESS

Directory users (persons or application processes) must access the Directory through *Directory User Agent* (DUA) which converses with the Directory using the *Directory Access Protocol* (DAP). When we relate the Directory Access Model in Figure 8 to the SCF Model in Figure 3, it is easy to see that the SCF Data Access Manager is the DUA and the Service Logic Processing Program Instances are the Directory users.

The DAP allows the DUA to issue requests without waiting for the completion of previous requests. Before an DUA can issue any requests, it must establish an association to the Directory. It should be noted that it is not specified in X.500 Series of Recommendations how the DUA obtains a presentations address for the Directory. During the establishment the authentication takes place. After the Directory has validated the user's identity, the DUA may issue requests; *queries* to receive information in the DIB and *updates* to modify either the structure or contents of the DIB.

The Directory provides four types of query operations: `Read`, `Compare`, `List`, and `Search`. `Search` operation will be in the subset of X.500 that is adapted in the next CS-1 Refinement. `Search` operation has four parameters: base_object, subset, filter, and selection. The base_object specifies the object from which the search begins. The subset gives the scope of the search in terms of entries. The filter gives the criteria that is used to eliminate entries. The selection specifies the attributes of the filtered entries which are returned as the result of the operation.

There are four types of update operations: `AddEntry`, `RemoveEntry`, `ModifyEntry`, and `ModifyDN`. The `ModifyEntry` will be in the next CS-1 Refinement, whereas the use of `AddEntry` and `RemoveEntry` operations is still under further study. The `ModifyEntry` has two arguments: object and changes. The object gives the name of the entry that is modified. The changes gives a list of changes: add attribute or value, remove attribute or value.

4.3 STRUCTURE OF THE DIRECTORY

When the IN services are globalized, the data hold in the the Directory can be viewed as a world-wide logical entity, into which many different organizations may provide entries. The functional model of the Directory is shown in Figure 9. The Directory consists of processes called *Directory System Agents* (DSAs) which communicate with one another using the *Directory System Protocol* (DSP). When we relate the Functional Model of the Directory in Figure 9 and the functional entities on the distributed functional plane of the IN conceptual model given in Figure 1, we see that an SDF is a DSA. This resolves the unspecified interface between the SDFs.

Suppose that a DSA has received a request that it cannot complete. The DSA has now two options. Firstly, it can communicate with other DSAs on the behalf of the DUA to complete the request. Secondly, it can simply *refer* the DUA to a DSA that it believes will be better able to complete the request. The refer option implies that each DSA must maintain a set of *knowledge references*. Since each DSA contains a part of the DIT, the DSA must know the presentation addresses of those DSAs which contain the immediate parents and children of its entries. The X.500 Recommendations specify the protocols that are used to create, maintain, and extend the knowledge held by each DSA. These protocols are not in the scope of the CS-1 Refinements but they should be studied for the management aspects of the future IN Capability Sets.

In the Directory exactly one DSA is responsible for the creation and maintenance of an entry. This DSA is called the *master DSA* for this piece of information. The Directory allows that other DSAs may have copies of some pieces of information. To simplify the implementation of the DSAs, the updates to the master information need not be synchronized with updates to its copies. As a result, the DIB may, at times, be in an inconsistent state. This is not a serious deficiency, since the users have always the option of requesting access to the master information.

The 1992 Amendments to X.500 present two different mechanisms for replication information: *caching* and *shadowing*. Caching occurs when a DSA stores locally a copy of

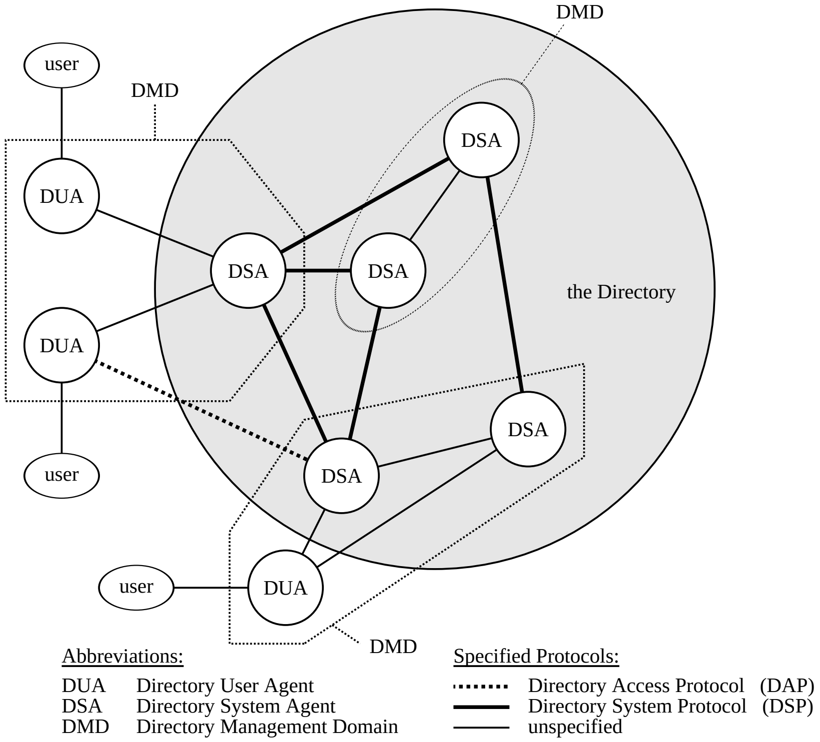

Figure 9: Functional Model of the Directory

an entry used to fulfill a request. This may result in a substantial performance improvement if the same entry is rerequested. Shadowing occurs when one DSA mirrors some information held by another DSA. The shadowed information is brought into agreement with the master information on some predetermined schedule.

The replication mechanisms of X.500 need further study. We need to examine the semantics of correctness in transaction processing. Do we need strict serializability or can we relax it? Another topic for further study is how the shadowed information should be updated. Finally, we are interested in extending the caching. In some situations, the nearest DSA/SDF is still too far away from the DUA/SCF. We want to learn if it is feasible that the DUA/SCF caches some very frequently requested entries.

4.4 ADMINISTRATION AND SECURITY

In addition to attributes that hold information about single facets of an object, there are attributes that hold administrative information, e.g., knowledge references and access control items. Such attributes are known as *operational attributes* that are normally invisible to ordinary users. Operational attributes that apply to more than one entry are stored in a special kind of entry known as a *subentry*[5]. Each subentry has a

[5]Besides storing administrative information, subentries can also be used to store attributes shared by more one entry.

`subtreespecification` attribute that specifies the set a entries to which the remaining attributes apply.

Access control to the information in the DIB can be based on an entry, on an attribute, or on an attribute value. Two methods for verifying user's identity have been specified: *simple* and *strong authentication*. Simple authentication is based on passwords while strong authentication is based on public key cryptography; for details, see e.g. Bumbulis et al. [1993], Burrows et al. [1990], Fumy and Leclerc [1993].

The needs for authentication in IN services should be carefully analyzed and evaluated. Intuitively, we believe that simple authentication is sufficient for most of queries that need authentication. Many queries may be processed without any authentication. However, some queries, most updates and all access to operational attributes may require strong authentication.

5. MANAGEMENT ASPECTS

The *Telecommunication Management Network* (TMN) is a generic architecture to be used for all kinds of management services. It is based on the principles of the OSI Management (X.700 Series of Recommendations) and is standardized in M.3000 Series of Recommendations. The integration of IN and TMN architectures is one of the most important goals in the IN long-term architecture. The reason is that it will be impractical to support two independent architectures (TMN and IN) when applications on both architectures must interoperate.

In this section we briefly summarize IN Management, TMN and OSI Management. We concentrate on the aspects that are important in the database access. Other aspects can, for example, be found in Appeldorn et al. [1993] and in Yemini [1993]. As we have already noticed in the Perspective, the database has a twofold functionality. Firstly, the database and its elements are managed objects. Secondly, the database may be used to implement the *Management Information Tree* (MIT) which is a focal point in TMN (and in OSI Management).

5.1 IN MANAGEMENT AND TMN

The management aspects of IN are primarily concentrated on service management. On the distributed functional plane of the IN conceptual model there are three functional entities related to management. These are the *Service Management Access Function* (SMAF), the *Service Management Function* (SMF), and the *Service Creation Environment Function* (SCEF). Figure 10 shows the management control relationships between these FEs, the SCF, and the SDF. As the figure indicates, the SCF and SDF only need a management interface to the SMF. It should be noticed that the management interfaces have not yet been specified. However, the integration of IN and TMN would lead to the usage of a TMN-type stack of protocols including ACSE, ROSE, and CMISE (*Common Management Information Service Element*).

The SMF handles five groups of activities: service deployment, service provisioning, service control, billing, and service monitoring. In each group there are activities that are related to data manipulation. Service deployment includes allocation of service generic data. Service provisioning includes introduction and allocation of customer specific data.

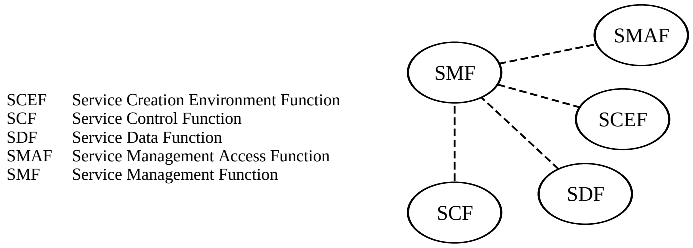

Figure 10: Some Functional Entities and Management Control Relationships

Service control contains updating service generic and customer specific data. Billing includes generating and storing charging records, collecting charging records, and modifying tariffs. Service monitoring includes collecting measurement data.

TMN has identified several *TMN Management Services* that are built on a set of *TMN Management Service Components*. The components, in their turn, are based on *TMN Management Functions*. The TMN functional architecture is described through *function blocks* which are similar to IN functional entities. The function blocks identified in TMN are the *Network Element Function* (NEF), the *Operations System Function* (OSF), and the *Work Station Function* (WSF).

The NEFs model all entities that form the network to be managed. Each NEF should physically locate on the network element it manages. The WSFs represents the functionalities and information that relate to the man-machine communication in TMN. The OSFs include the functions for processing, storing, and retrieving management information. TMN has identifies four different OSFs that correspond to the business, service, network, and element management layers. These OSFs are known as B-OSF, S-OSF, N-OSF, and NE-OSF.

5.2 OSI MANAGEMENT

The fundamental idea in the OSI Management is that the knowledge representing the information used for management is separated from the functional modules performing the management actions. This can be seen in the information and functional aspects of the systems management model. Another important feature of OSI Management is distributed processing. Since the environment to be managed is distributed, the individual components of the management activities are themselves distributed.

As shown in Figure 11 the interactions that take place are abstracted in terms of *management operations* and *notifications*. Management activities are effected through the manipulation of *managed objects* (MOs). An MO is the abstraction of a resource that represents its properties as seen by management. An essential part of the definition of a managed object is the relationship between these properties and the operational behaviour of the resource.

Systems management application entities (MIS-users) can take either manager role or agent role. An agent manages the MOs within its local system environment. It performs management operations on MOs as a consequence of management operations issued by

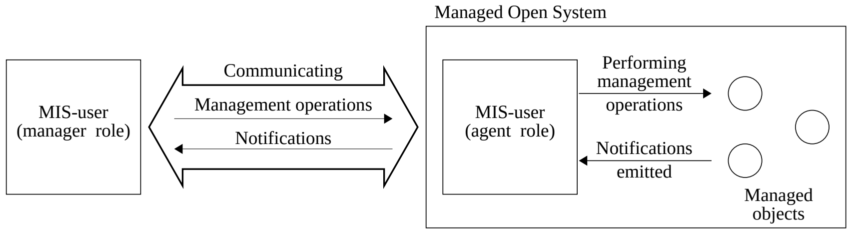

Figure 11: Systems Management Interactions (Figure 1/X.701)

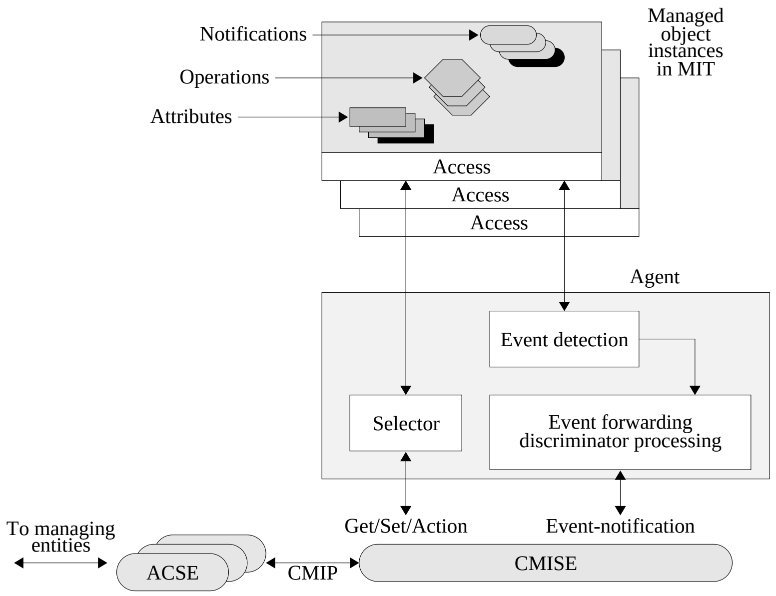

Figure 12: Agent Communication Architecture in OSI Management

a manager. An agent may also forward notifications emitted by MOs to a manager. The agent maintains a part of the *Management Information Tree* (MIT), which is a dynamic database. The MIT contains instances of MOs organized on a hierarchical database tree, similar to the DIT in X.500 Directory.

Agents and managers are viewed as peer applications. They exchange information using the services of CMISE. CMISE uses the *Common Management Information Protocol* (CMIP) and utilizes the services of ACSE and ROSE. Figure 12 exemplifies the structure of an agent communication architecture.

When we relate the OSI Management to the SDF and SMF, the agent corresponds to the Maintenance Functions in the Functional Model of SDF Application Entity given in Figure 6. The SMF corresponds to the manager. This is for the SDF management. Another part of the SDF-SMF interactions is the data manipulation needs of the SMF. In billing the SMF stores charging records. Now the SDF can provide the database for those records. The MIT is not necessarily the best way of organizing the charging database. A traditional database based on the relational model can be better suited to such data handling. However, this increases the complexity of SDF.

6. SUMMARY

We have examined current ITU-T Recommendations in Q.1200, X.500, and X.700 Series which are relevant to database access in the IN architecture. The database must implement the Directory Information Tree and the Directory System Agent. In addition, the database must provide User Data Manipulation ASE with Query and UpdateData operations in order to be compatible with the CS-1 INAP. Moreover, the maintenance functions of the SDF must implement the OSI Management Agent (or NEF function block of TMN). This implies that the SDF must include CMISE. It must also maintain the dynamic Management Information Tree database. The future IN Capability Sets, perhaps CS-3, may require additional database capabilities supporting SMF data handling needs.

In addition to the requirements above, the implementation of the database system to be used in IN calls for further study in various research areas. The research topics include: real-time transaction processing, distributed main memory databases, object-oriented data models, relaxed serializability for correctness of transaction, performance and reliability evaluation methodologies for real-time databases. These questions are examined in the Darfin[6]-project. During the academic year 1994/5 the project is funded by Telecom Finland.

REFERENCES

Appeldorn, M.; Kung, R.; and Saraccor, R. (1993) "TMN + IN = TINA". *IEEE Communications Magazine* **31**, 3 (Mar.) 78–85.

Bumbulis, P. J.; Cowan, D. D.; Durance, C. M.; and Stepien, T. M. (1993) "An Introduction to the OSI Directory Services". *Computer Networks and ISDN Systems* **26**, 2 (Oct.) 239–249.

Burrows, M.; abadi, M.; and Needham, R. (1990) "A Logic of Authentication". *ACM Transactions on Computer Systems* **8**, 1 (Feb.) 18–36.

Cattell, R. G. G.; ed. (1994) *The Object Database Standard: ODMG-93*. Morgan Kaufmann, San Mateo, Calif.

Chatras, B. and Gallant, F. (1994) "Protocols for Remote Data Management in Intelligent Networks CS1", In *Workshop Record of IEEE Intelligent Network '94 Workshop*, Volume 1. (Heidelberg, Germany; May 24–26, 1994).

Fumy, W. and Leclerc, M. (1993) "Placement of Cryptographic Key Distribution Within OSI: Design Alternatives and Assesment". *Computer Networks and ISDN Systems* **26**, 2 (Oct.) 217–225.

OMG Object Model (Draft), May, 1992. OMG TC Document 92.5.1.

Raymond, K. A. (1993) Reference Model of Open Distributed Processing: A Tutotial. In *Procedings of the Intenational Conference on Open Distributed Processing* (Berlin, Germany; Sep. 13–16, 1993).

Soley, R.; ed. (1993) *Object Management Architecture Guide*, Second Revision. Object Management Group, Framingham, Mass.

Sykas, E. D. and Lyberopoulos, G. L. (1991) "Overview of the CCITT X.500 Recommendations Series". *Computer Communications* **15**, 9 (Nov.) 545–556.

Yemini, Y. (1993) "The OSI Network Management Model". *IEEE Communications Magazine* **31**, 5 (May) 20–29.

[6]Database ARchitecture For Intelligent Networks