Recent Developments in Telecommunications Software Architectures

Prof. Kimmo E. E. Raatikainen

Department of Computer Science, University of Helsinki

P.O.Box 26 (Teollisuuskatu 23), FIN-00014 University of Helsinki, Finland

E-mail: kimmo.raatikainen@cs.helsinki.fi

Abstract

During the last ten years telecommunications markets have met significant changes in Europe and USA. Old PTT monopolies are disappearing and open competition is already a fact. Another significant trend is the merging of broadcasting, telecommunications, and data communications. The consequence is that new services must be timely developed and provided. Telecommunications software architectures are considered as the enabling means of satisfying these new requirements. This paper gives an introduction to two such architectures: Intelligent Network (IN) and Telecommunication Information Networking Architecture (TINA).

1 Introduction

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 modeling, information technology, cooperative processing, distribution control, management of services and networks, verification/validation, and artificial intelligent.

TINA–C, a world–wide consortium developing a so–called Telecommunication Information Networking Architecture, has the goal to define and validate an open framework for integrated control and management of future telecommunications services and resources. The framework is based on distributed computing, object orientation, and other standards or recommendations in the fields of telecommunications and information technology, namely: Open Distributed Processing (ODP), Common Object Request Broker Architecture (CORBA), Intelligent Networks (IN) and Telecommunications Management Network (TMN). However, whereas ODP and CORBA are substantially adopted as they are, IN and TMN are only taken as reference legacy in view of a synthesis overcoming both and resolving the current dichotomy between control and management planes.

2 Intelligent Network

2.1 Perspective

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.

The need for interoperability calls for openness, particularly for open distributed processing. From the computational viewpoint[1], the services

[1]Viewpoints are pragmatic tools defined in the ISO *Reference Model of Open Distributed Processing* (RM–ODP). Each viewpoint—enterprise, information, computa-

and the computing platforms can be represented as an aggregation of collections of objects. This object–oriented modeling 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 modeling, 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 OSI–

tional, 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.

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

2.2 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*

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

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 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.

2.3 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:

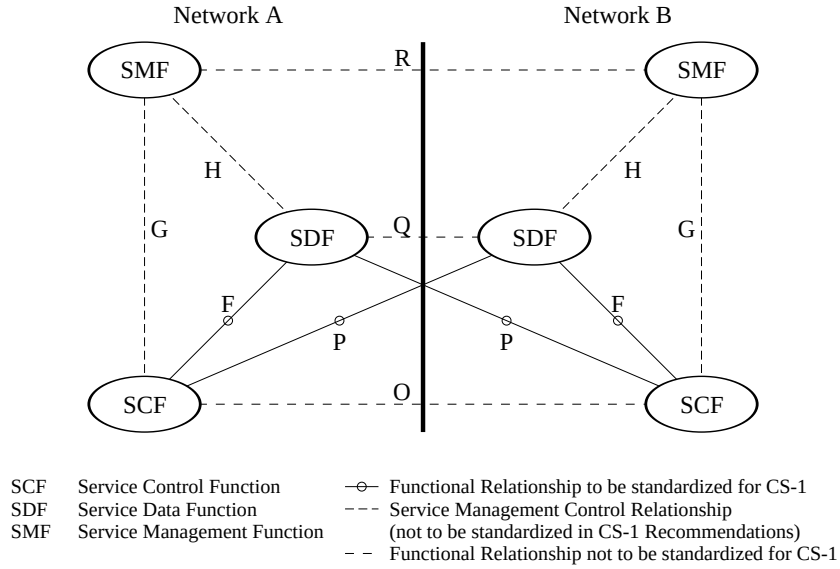

Figure 1. Functional Relationships and Reference Points for CS–1

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 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.

2.4 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

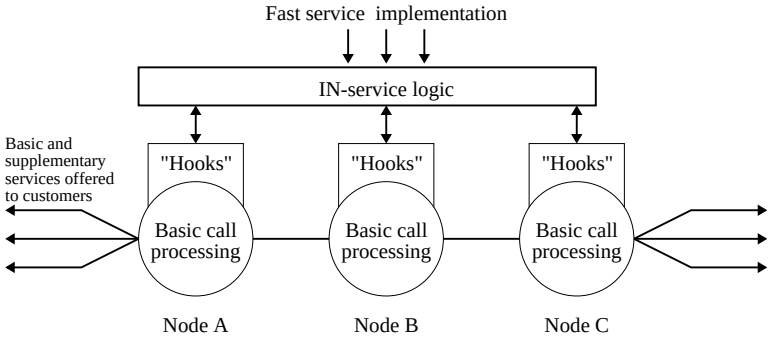

Figure 2. IN service processing model (Figure 16/Q.1201)

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 o a query to a separate SDF.

In Recommendation Q.1214 *Distributed Functional Plane for Intelligent Network CS–1* 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).

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

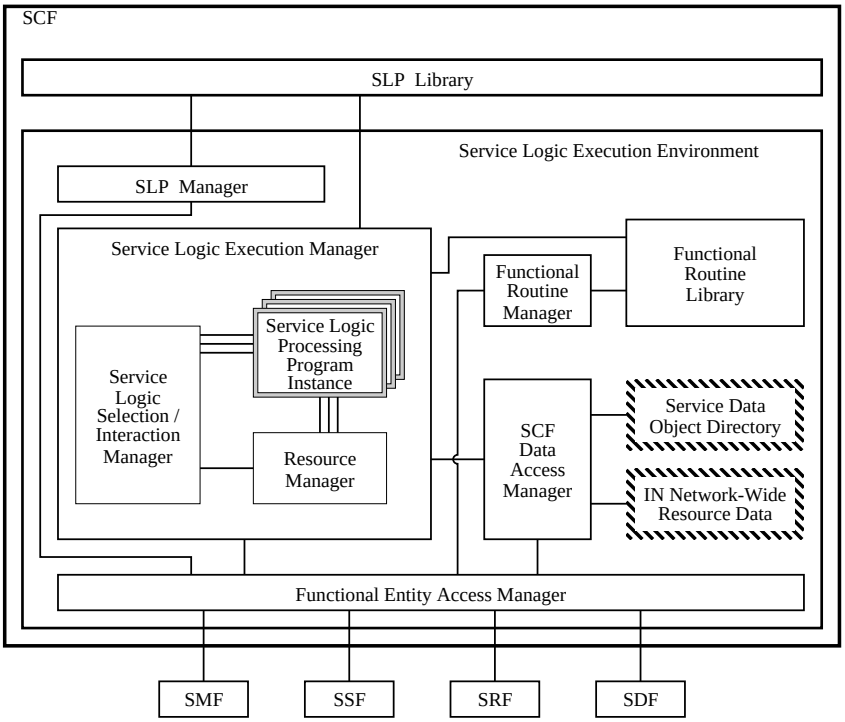

Figure 3. Service Control Function Model
(Figure 4-19/Q.1214)

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 life time 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 message 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,

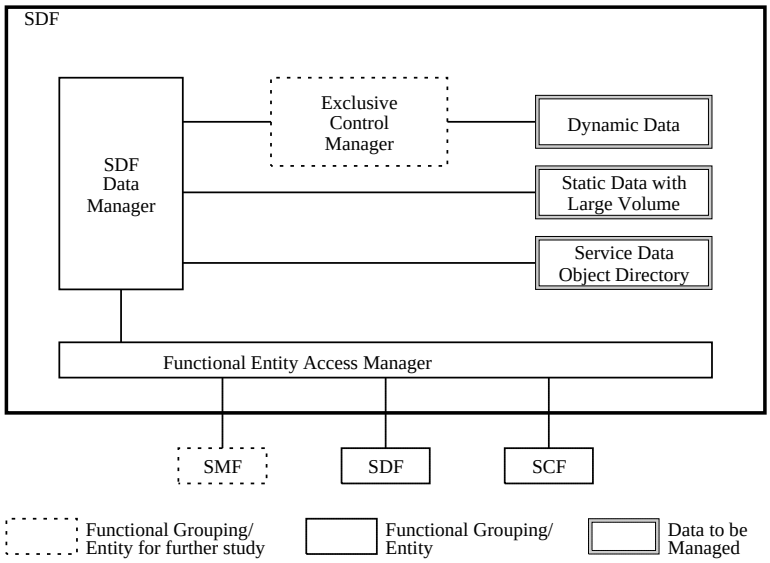

Figure 4. Service Data Function Model (Figure 4-20/Q.1214)

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.

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 dif-

ferent 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 dialed 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 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.

2.5 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 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.

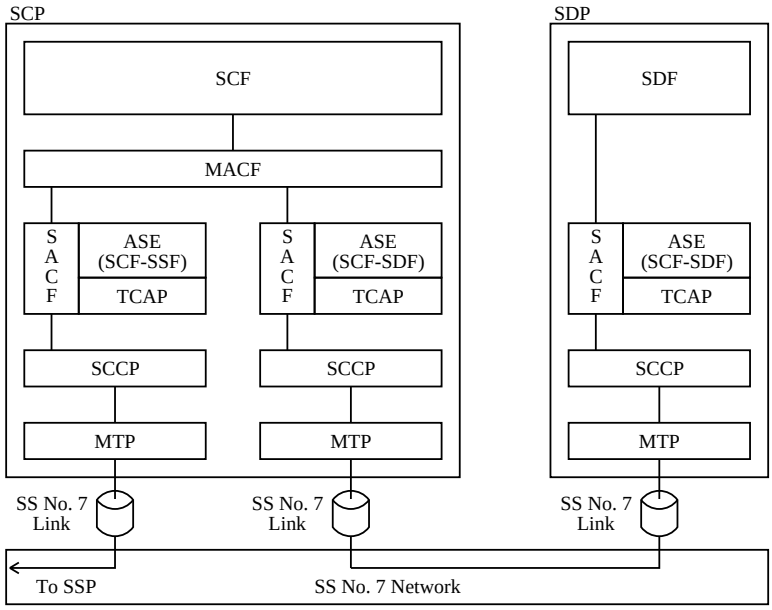

Figure 5. Physical Interface Between SCP and SDP
(Figure 1/Q.1218)

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

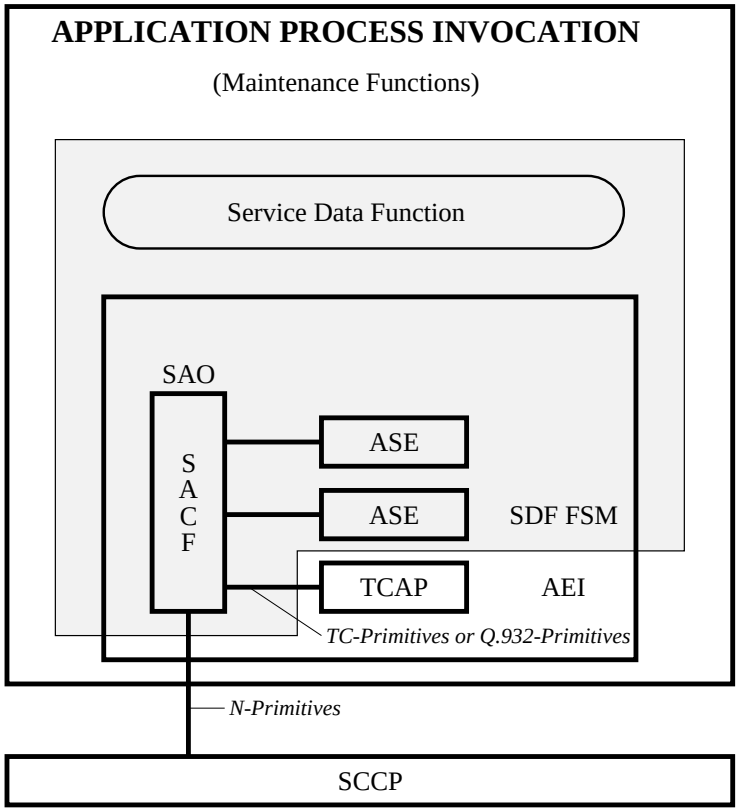

Figure 6. Functional Model of SDF Application Entity
(Figure 41/Q.1218)

operation Return Result from Query and UpdateData, respectively.

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,

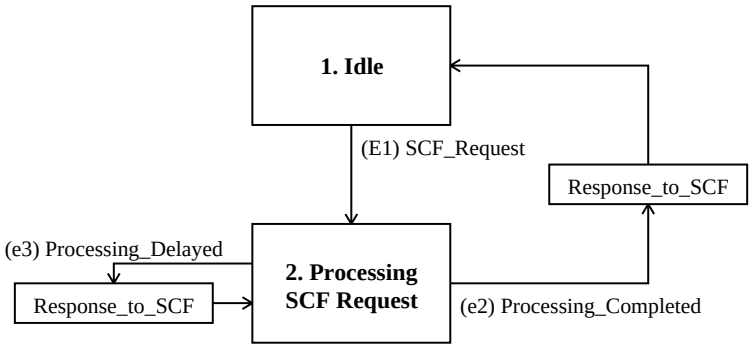

Figure 7. The SDF Finite State Model (Figure 42/Q.1218)

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 Telecommunication Information Networking Architecture

The essential elements of the TINA, that is the *Telecommunication Information Networking Architecture*, framework include a business model, a network resource architecture, a service architecture, and a connection management architecture. Below we briefly summarize each of them.

However, we start by a brief introduction to the concepts of Service Architecture.

3.1 Overview of Service Architecture

The concept of Service Architecture was introduced in the very late 80's as a framework combining software architectures in telecommunications and information technology. A Service Architecture targets a software platform which is independent from network technologies and which enables open service provision (the term service platform is specifically used, see below). An open service creation environment may also be built in conformity with a Service Architecture. A service architecture essentially separates teleservices from bearer services. It adopts a long–term vision that has three levels (see Figure 8):

- Resource Infrastructure that includes network technologies and supporting elements like protocols, interfaces, and databases.
- Service Platform that includes all software entities involved in service creation, provision, and management.
- Application Stratum that includes all elements specific to applications.

Going further, three logical blocks can be identified within the Service Platform:

- *Service Execution Environment* that provides a basic set of open distributed processing services,
- *Service Control and Management Components* that support various applications in their communications needs, and
- *Resource Adapters* that encapsulate resource dependent features and provide a uniform way to access, control, and manage elements in the Resource Infrastructure ('uniformity' with the respect of the variety of manufacturer products in the same class of functionality, for instance ATM switches).

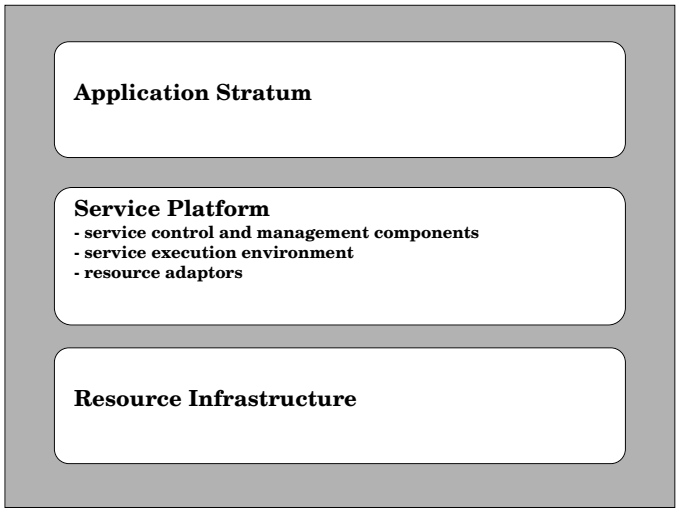

Figure 8. Long–term Vision in Service Architectures

3.2 TINA Business Model

The business model defines a set of concepts and guidelines to specify reference points, which in turn specify conformance requirements, in a TINA system. The business model starts by identifying a set of business roles, i.e. possible functions covered by any telecom actor in relation to another, in a deregulated, multi–provider, multi–domain environment:

- Consumer, defined as the role using the services provided by a TINA system. A consumer can use services, after subscribing from one or many retailers (see below).

- Retailer, defined as the role serving any stakeholder in its consumer role. The retailer provides a "supermarket" or a "boutique" to consumers, possibly in a one–stop fashion, by resorting to other providers (brokers, third party service providers, and connectivity providers, see below) to purchase the elements needed to ensure the desired services to its consumers.

- Broker, defined as the role providing information to actors that enables them to find other actors and services in the TINA system.
- Third party service provider, defined as the role supporting retailers or other third party service providers with services. Unlike a retailer, a third party service provider does not have any contractual relationship with consumers.
- Connectivity provider, defined as the role which owns and manages a network, be it with access, transport, or signaling functionality.

3.3 TINA Network Resource Architecture

The network resource management architecture provides a model to deal with the underlying network resources as seen by the Service Architecture (described below). Here only the essentials are reported. The concept of *layer network* is used to represent a particular network technology (for instance ATM, GSM or DCS). The concept of *subnetwork* is used to represent the domain of resources belonging to a layer network within the management scope of a single stakeholder. An overall path from one consumer terminal to another, involving the traversal of a number of layer networks, each possibly resulting from a concatenation of subnetworks, is said *trail*. The concept of tandem connection is used to denote any continuous segment of a trail outside the management scope of a given connectivity provider.

3.4 TINA Service Architecture

The TINA Service Architecture is a set of concepts, rules, guidelines and recipes to model services, their control and management. Key to the Service Architecture is the concept of session, which is of relational nature. A session is a temporary association among a group of objects established in order to fulfill a common task or objective over a time duration. In more concrete terms, a session represents a collection of resources and logic required to provide an instance of a service.

Three different kinds of sessions are foreseen in the Service Architecture and a fourth one in the connection management architecture (see Figure 9):

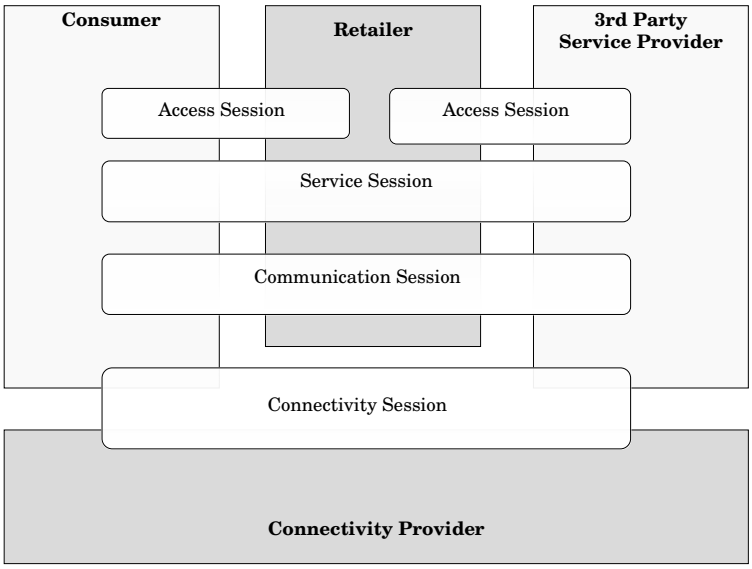

Figure 9. Sessions and Business Roles in TINA

1. An access session is an environment for granting a consumer a trusted association with a retailer. It covers the objects and their relations/interactions preliminary needed for creation and management of all service sessions (see below).

2. A service session is an environment for achieving specific service goals. It contains both generic (common to all services) objects and specific (service dependent) objects. Since each service has a provider and a consumer, the service session can be divided into a provider part and a consumer part.

3. A communication session is an environment for establishing and managing a number of related connections meaningful at service level. It is used to create and maintain a number of connections at the level of stream flow. The communication session supports the service provider (a retailer or a third party service provider) part of the connection management. A communication session is a user of

the connectivity session, associated with a connectivity provider, to establish actual communications.

4. A connectivity session is an environment for establishing and maintaining network connections. It is used to create and maintain a number of connections at the level of network flow. The connectivity session supports the connectivity provider part of the connection management. It provides a technology independent view of network connection. The connectivity session maps this abstract view onto a technology dependent view offered by the supporting layer networks.

3.5 TINA Connection Management Architecture

The TINA Connection Management Architecture foresees a *Network Management Layer*, an *Element Management Layer* and a *Network Element Layer*. The computational objects in the Network Management Layer are grouped into four sublayers:

- Communication Session sublayer:
  - Communication Session Manager Factory[4] (CSMF),
  - Communication Session Manager (CSM), and
  - Terminal Communication Session Manager (TCSM) objects.
- Connectivity Session sublayer:
  - Connection Coordinator Factory (CCF),
  - Connection Coordinator (CC), and
  - Flow Connection Controller (FCC) objects.
- Layer Network sublayer:
  - Layer Network Controller (LNC),
  - Trail Manager (TM),
  - Tandem Connection Manager (TCM), and
  - Terminal Layer Adapter (TLA) objects.

[4]The term 'factory' denotes a permanent object, whose main mission is to create other objects possibly existing only for the lifetime of a session or a connection.

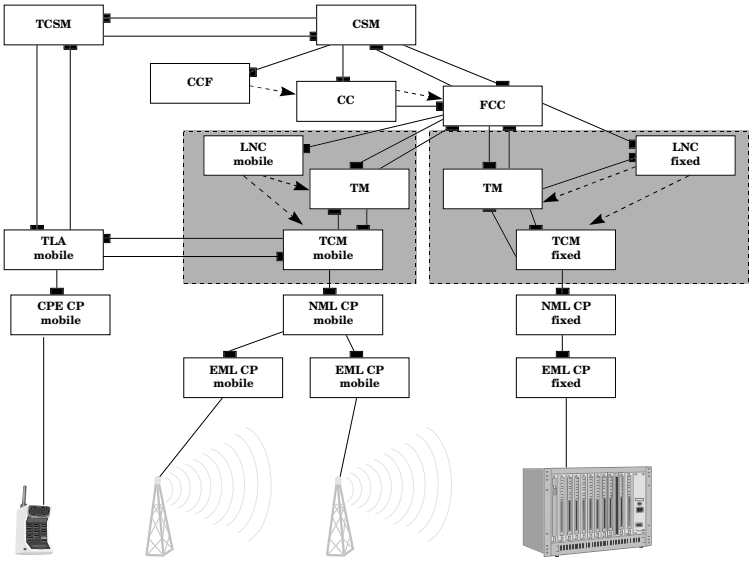

Figure 10. Computational Overview of TINA Connection Management for Mobile Terminals

- Subnetwork sub–layer:
  - NML Connection Performer (NML CP) and
  - Customer Premises Equipment Connection Performer (CPE CP) objects.

In the Element Management Layer there are EML Connection Performers (EML CPs) and in the Network Element Layer there are Network Elements.

Figure 10 shows a specialization[5] of the TINA connection management model displaying the computational objects involved when a mobile terminal is connected to a fixed network.

[5]Elaborated in the ACTS DOLMEN Project.

Literature

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

[2] *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.

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

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

[5] *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).

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

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

[8] *Raymond, K. A.* (1993) Reference Model of Open Distributed Processing: A Tutorial. In Proceedings of the International Conference on Open Distributed Processing (Berlin, Germany; Sep. 13–16, 1993).

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

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

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