Database usage and requirements in intelligent networks

J. Taina and K. Raatikainen
University of Helsinki, Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23), FIN-00014 University of Helsinki, Finland.
Telephone: +358 9 7084 {4247,4243}. Fax: +358 9 7084 4441.
E-mail: {juha.taina,kimmo.raatikainen}@cs.Helsinki.FI

Abstract

In this paper we present and analyze IN data managers for dynamic data. We examine database needs for call control, service switching, special resources, service data, and service management. We take a closer study for database requirements for service data in SDF and management data in SMF. The issues addressed in the requirements analysis are needed application interfaces, logical object model, and core database requirements. From the analysis we draw a conclusion that it is possible to fulfill the requirements and that SDF and SMF can share the same database architecture if compromises are made.

Keywords

IN CS-1/2, database interfaces, logical data models, performance, fault-tolerance, data security

1 INTRODUCTION

Databases in Intelligent Networks have gained a dominant role in the last few years. Not only service-related data is in databases, but also special resources, management, switching, IN service triggering, and network traffic control have data that can be managed in a database management system.

The nature of the database managers differ drastically. Service management must support fast data access that is tailored for reads while management databases should be tailored for reliable distributed updates. Service switching and IN triggering need very fast data access with little security and transaction control. Special resource functions need multimedia data services. All these classes have different database needs. An open question is: how much the database requirements of different managers have in common?

In this paper we try to answer that question. In order to do that, we need to identify the data managers in Intelligent Networks and analyse their characteristics. After that we can summarise the requirements that are set for the managers, and how they can be met.

The requirements analysis for database architectures in IN is derived from the work that has been done in international standardisation, especially in ITU-T. From database management point of view, most interesting part of IN is the IN Distributed Functional Plane (DFP) ITU-T 1993a, 1996), in particular the recommendations in the Capability Sets 1 and 2 (IN CS-1 and IN CS-2). In the recommendations, the database requirements are given as functional description of the elements. Database requirements for telecommunications based on IN CS-1 have been studied in Eurescom research projects (Eurescom 1993, 1994). Based on the Object Distributed Processing (ODP) engineering viewpoint, the database requirements have also been examined (Kerboul *et al.* 1993).

The rest of this paper is organised as follows. In Section 2 we summarise functional entities in CS-2 Distributed Functional Plane (DFP) that are either clients or servers for database requests. In Section 3 we take a closer study of the requirements of SDF and SMF data managers.

2 DATABASE ASPECTS IN INTELLIGENT NETWORKS

Databases have a dominant role in IN CS-2 Distributed Functional Plane (DFP). Not only the Service Data Function has a data management, but also several other functional entities have either database managers or services that need database management. In the CS-2 architecture, CCF, SSF, SRF, SDF, and SCF has data managers. The SMF architecture is left for further study in CS-2. Since management needs to keep track of managed data, a database management system is needed in the SMF architecture.

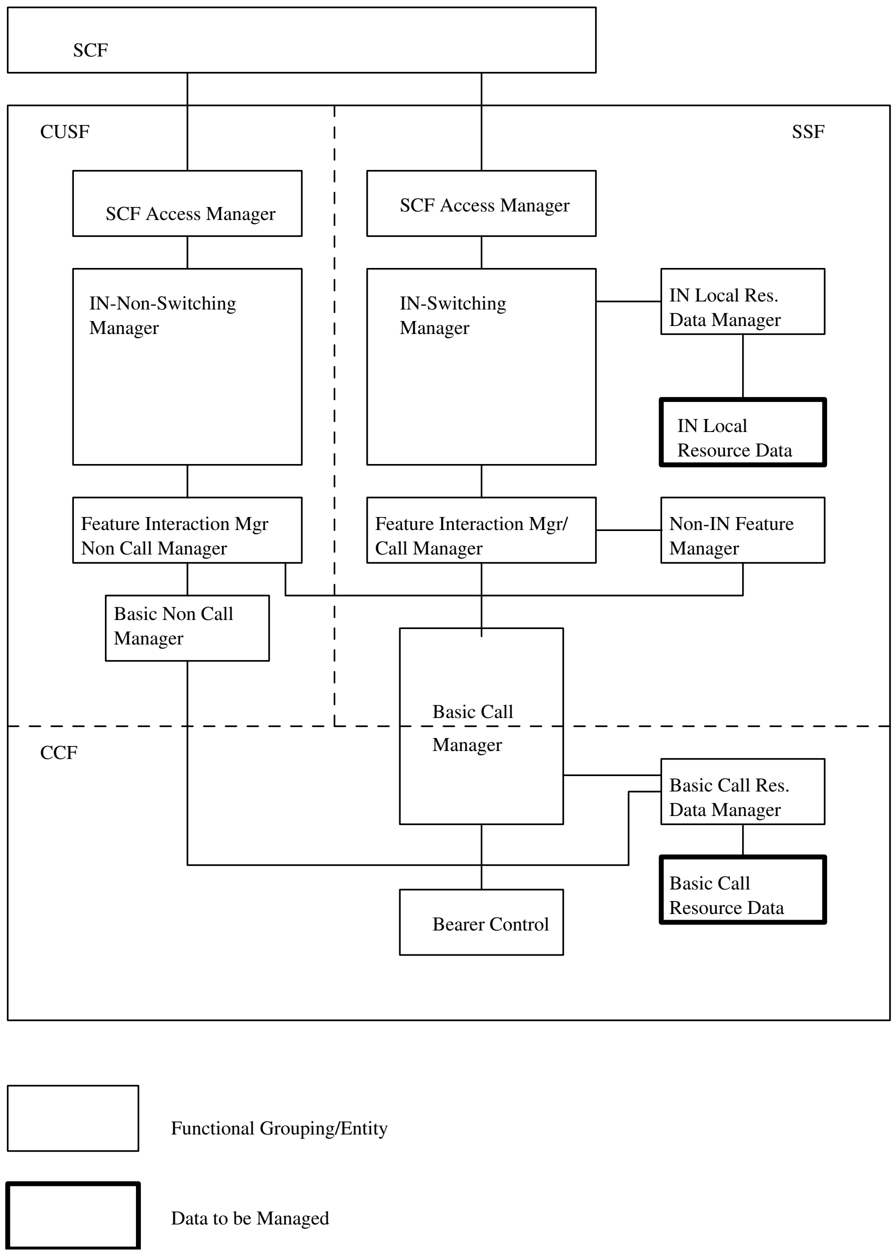

Figure 1 Q.1224/IN CS-2 service switching/call control function architecture.

Most of the data managers are embedded to functional entities. Only data managers in SDF and perhaps SMF are global in the sense that they can accept requests from other functional entities. The managers of other entities service requests from inside the entity.

2.1 Service switching and call control functions

The functional model of SSF/CCF/CUSF is described in Figure 1. The most interesting functional entities in the model are two data managers: IN local resource data manager and Basic call resource data manager. Neither of the managers have been described in CS-1, except that they are assumed to exist (ITU-T 1993a). The CS-2 draft only states that CCF manages basic call resource data which consists of call references (ITU-T 1996). Hence it is difficult to say what are the requirements for the data managers. We assume that the IN local resource data manager is responsible of managing data that is related to triggering IN services. Such data has a simple structure. A manager must be tailored for fast data access with a simple key. As of the Basic Call Resource Data Manager, we assume that it is responsible of managing dynamic call connection data. The managed data is temporal. Static data may be present if routing information is kept in the database.

2.2 Special resource function

The Special Resource Function (SRF) offers specialised resources for IN services that are beyond Service Control Function services. For instance, SRF is responsible of voice recognition, conference bridges, etc. As such, it is a very specialised functional entity with its own database requirements.

The architecture of CS-2 SRF is in Figure 2. The most interesting entity of the architecture is the Data Part (DP). The ITU-T Q.1224 draft states that the DP is composed of Database manager and database which contains recorded voice, sound, image, text (ITU-T 1996).

The SRF Data Part is a multimedia database. As such, it has to offer fast data access for complex data. The database is defined to be a server for internal data requests. The SRF can also be a client for a SDF since it can accept service logic scripts from SCF that can include SDF data requests. The SRF management has been left for further study in CS-2. The recommendations state that SMF is responsible of management of the service specialised resources, such as User Interaction scripts, resource functions, and data. It is also possible that service subscribers can manage their private data in SRF. Together these request types imply that the SRF must offer a database interface for external requests.

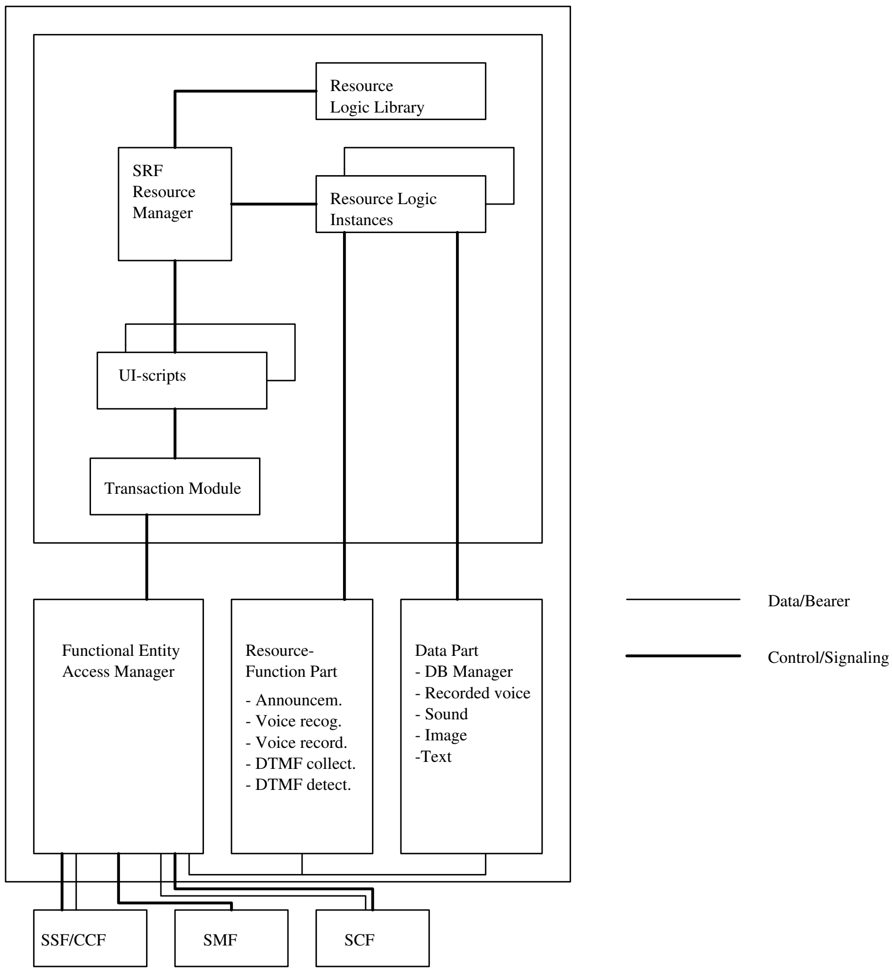

Figure 2 Q.1224/IN CS-2 service resource function architecture.

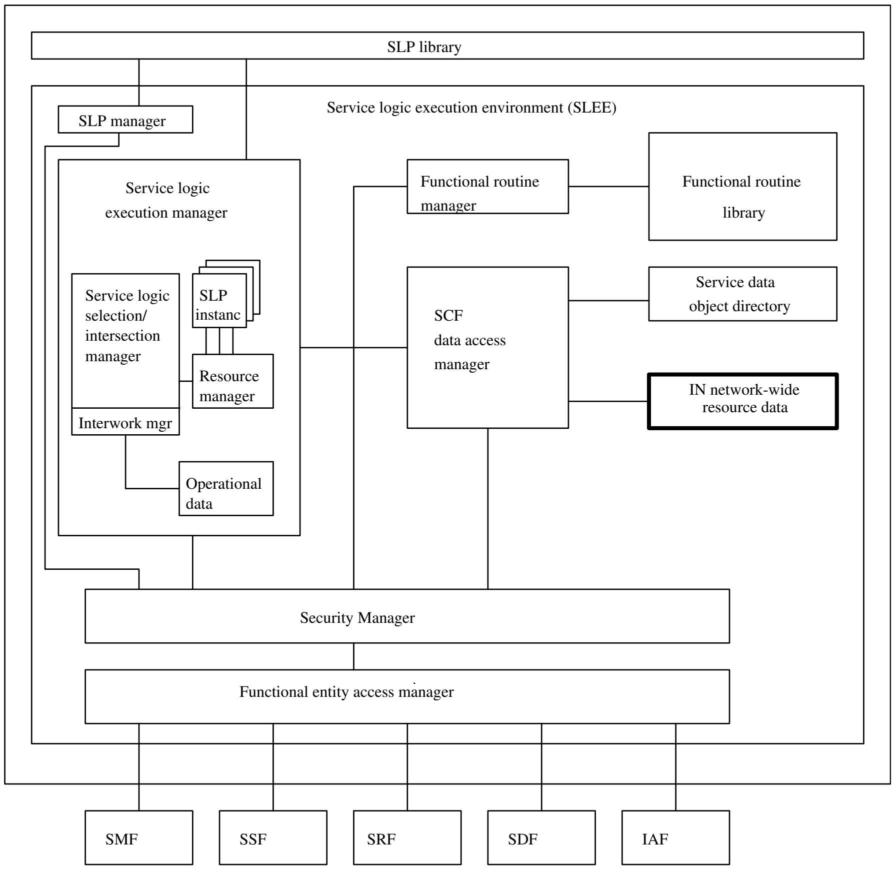

Figure 3 Q.1214/IN CS-1 service control function architecture.

2.3 Service control function

A similar structure than in SSF/CCF/CUSF is present in CS-2 SCF model in Figure 3. The architecture model has two data managers: Resource manager and SCF data access manager. The resource manager controls the local SCF resources and provides access to network resources in support of service logic program execution. The former resources are needed only on SCF execution and can be considered an integrated member of the architecture. The latter resources, while referencing to global network related data, are accessed via the other data manager, SCF data access manager. Thus, the resource manager sets requirements only to the SCF data access manager and not directly to an IN database architecture.

The SCF data access manager provides the functionality needed to provide for storage, management, and access of shared and persistent information in the SCF. The data access manager also provides the functionality needed to access remote information in SDFs (ITU-T 1996). This implies that the data access manager is both a client to one or more SDFs, and a database manager for SCF local data.

The SCF data access manager manages two types of data: the Service data object directory and the IN network-wide resource data. The Service data object directory provides means to address the appropriate SCF for access to a specific data object (ITU-T 1996). This implies that the data object directory is used for SCF interaction instead of SDF data requests. However, the recommendations also state that the service logic element manager, which is responsible of service logic programs, interacts with the data access manager to access service data objects in SDFs. The SCF data access manager uses the service data object directory to locate service data objects from SDFs (ITU-T 1996). This implies that the service data object directory is used for SDF access. We assume that the SCF data access manager is responsible for all remote data access regardless of its location. It can access both SCFs and SDFs.

The other data element, IN network-wide resource data, is defined to be a data base for information about location and capabilities of resources in the network that can be accessed by the executed service logic programs (ITU-T 1996). Furthermore the recommendation states that usually the addressed functional entity is a service resource function SRF. Hence the IN network-wide resource data is location information about useful service resources, mostly SRFs. A SCF requests special services from a SRF which executes the request. The SCF may forward a service script for execution or request a single operation (ITU-T 1996).

Our conclusion is that a SCF is a database client for SDFs and SCFs, and a service client for SRFs. The recommendations do not state what kind of data is stored in SCFs, but it is probably directly related to service execution. All global service-related data should reside in SDFs.

2.4 Service data function

The CS-2 SDF architecture model is in Figure 4. The entity is responsible of real-time access and maintenance of global network related data. The ITU-T recommendations state that the SDF contains and manages data which is related to service logic processing programs and accessed in the execution of the programs (ITU-T 1996). This definition limits SDF for a server of requests from SCFs.

According to the CS-2 recommendations, the SDF contains customer and network data for real time access by the SCF. Its tasks include secured data acquisition and management of data, distributed requests with data location transparency to the requester, inter-networking security, optimal distributed data access depending on the network traffic, data replication and access rights for

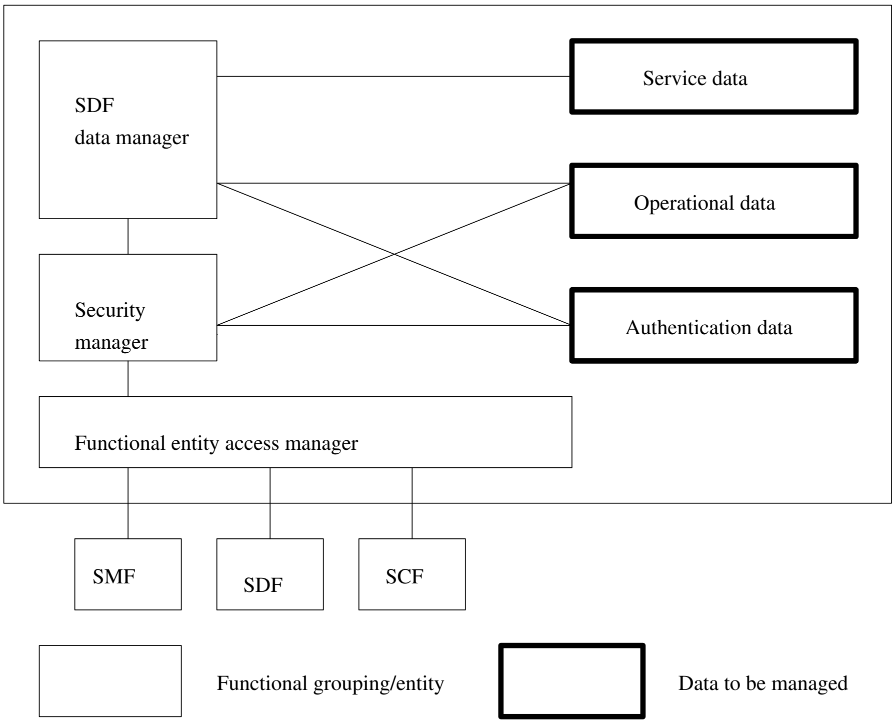

Figure 4 Q.1214/IN CS-1 service data function architecture.

replicated data, authentication and access control for secure access to service data, data support for security services, and fault-tolerance (ITU-T 1996). These tasks are typical to a distributed database management system.

The SDF data types are not as clearly listed. The recommendations state that SDF data types are authenticate data for user authentication and access rights (PIN-codes etc.), private operational data for SDF administration, and service data for the provision of a service. In other words, SDF data can contain anything. We have done a closer study of database use in CS-1 services and service features (Raatikainen 1994). There are five basic types of database operations that are needed in CS-1:

1. **Retrieval of structured objects from persistent subscriber data objects**. Some of the retrievals trigger a later update.
2. **User management actions that modify persistent subscriber data objects**. These actions must be protected against unauthorised use.
3. **Verification of Personal Identification Number** (PIN).
4. **Writing sequential log records**.

5. **Mass calling and Televoting**. The operations need high-volume small updates that must be serialised.

The most common of these operations are retrieving structured objects and writing sequential log records. Security operations, such as PIN verification, are also common. Management updates are not as common as the previous types. And finally, mass calling and televoting occur only on special occasions. They may also need SCFs that are specifically tailored for that use.

For IN CS-2, the third item (PIN verification) must be expanded to support general authentication and security services. Other than that, the list holds true in CS-2 as well.

The relationships between functional entities in IN DFP define how the entities communicate with each other. The Intelligent Network Application Protocol (INAP) that is required to support of Functional entity communication in CS-1 is defined in ITU-T (1993b). The INAP will support any mapping of functional entities into physical network entities. Thus the protocol is defined for maximal distribution, e.g. that every network entity consists of a single functional entity.

The INAP is a collection of Application Service Elements (ASE) that provide the necessary communication capabilities between functional entities. The CS-1 defines ASE for SCF-SDF communication is called "User Data Manipulation". It consists of two operations: Query and UpdateData. In Chatras *et al.* (1994) it is stated that the ASE is oversimplified. It has two problems: service-dependent semantics of operation parameters and authentication. The service-dependent semantics imply that the actions a SDF must perform are also service-dependent. The result of this is that every SDF must be tailored for each service. Authentication problems occur when a SDF is accessed from an external network. The SDF must perform the authentication of users and provide access control. In the ASE specification the only way to carry out authentication is to use attribute value comparison. The ASE does not define when the comparison should take place (Raatikainen 1993). In CS-2, the INAP interface exists for backward compatibility reasons. However, the recommended new SDF interface is based on X.500 Directory Access Protocol (DAP). The new protocol fixes the problems with ASE and tailored SDFs.

The SDF data management is not addressed in CS-2. Since SDF offers database management services, it should manage its own data. Write requests arrive mostly from SCFs and SDFs when service data is changing. In case of larger updates, such as adding new services and customers, the update request should arrive from the SMF. Hence SDF at least serves all types of SCF and SDF requests, and large SMF update requests.

2.5 Service management function

The SMF has been left for further study in CS-2. Here we assume that SMF will be based on Telecommunications Management Network (TMN) since one of the possible long term IN architecture plans is to be compatible with Telecommunications Information Networking Architecture (TINA). One of the goals in TINA consortium is to have a common foundation between IN and TMN (Demounem *et al.* 1995).

TMN (ITU-T Recommendations in M.3000 Series) is a generic architecture to be used for all kinds of management services. It is based on the principles of the OSI Management (ITU-T Recommendations in X.700 Series).

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. OSI Management is based on interactions between management applications that can take the roles of manager and agent. 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 agent manages the MOs within its local system environment. It performs management operations on MOs as a consequence of management operations issued by 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) that contains instances of MOs organised as a hierarchical database tree. In brief, the principles of OSI Management (and TMN) require that the database system contains the functionality of an OSI Management Agent.

If IN service management is based on TMN, the managed objects are all IN elements, including IN Distributed Functional Plane entities. The manager of a functional entity has knowledge of the management needs of the element. The SMF needs database services for maintaining this information. TMN architecture can be used for all network management, not only for IN element management. The more management is left for TMN, the more database services are needed.

3 REQUIREMENTS ANALYSIS

As we have seen, the CS-2 DFP has several database management systems. We classify them into two groups: embedded systems and external systems. An embedded database management system is an integral part of the functional entity where it is defined. An external database management system has clients at other functional entities, or it offers distributed database operations with other external systems.

The functional entities in CS-2 that have internal database management systems are CCF, SSF, and SRF. A CCF has Basic Call Resource Data manager, a SSF has IN Local Resource Data manager, and a SRF has Data Part. These database managers are so deeply integrated to the appropriate elements that their requirements are beyond the scope of a general analysis. In short, the SSF/CCF requires fast data access from databases because they are related to connecting calls and triggering IN services. The SRF data base is a multimedia database. The requirements of such a system are beyond normal analysis of a database management system. A comprehensive analysis of multimedia databases is in Khoshafian *et al.* (1996).

The IN Distributed Functional Plane has two external database management systems: Service Data Function, which has external clients, and Service Management Function, which must have distributed database features for management. Since CS-2 does not specify the structure of management data, we assume that it is based on the TMN data model.

In the analysis we consider the requirements from both SDF and SMF data management point of views. The issues addressed here are database interfaces and request types, logical data models, and core database issues. We have two goals for the analysis: 1) to identify the requirements and 2) to see how compatible the requirements are. If it turns out that SDF and SMF requirements are compatible, SMF can use SDF database services. In such a case, the SDF database is both a manager and a managed element. Even if SMF and SDF architectures are separated, the same design principles can be used on both architectures, and database management systems can access data from one another.

3.1 Database interfaces and request types

The needed database interfaces depend on the requests sources. A SDF has to answer requests that are related to service interaction. A SMF has to answer requests that are related to service management. These requirements also define the requirements for database interfaces.

A SDF may interact with SCFs, SDFs, and SMFs. Also in CS-2, it is mentioned that a SRF may request data from a SDF, for instance when a customer wants to use a different language in a message than the default one (ITU-T 1996). All these requests are local in a network. In addition, SDF has to answer requests from SDFs and SCFs in external networks, and requests that come from monitor terminals.

In CS-1, the SDF interface is CS-1 INAP. In CS-2, the defined SDF interface is X.500 DAP. Both of the interfaces must be supported in a SDF database. Next to these, a high-level query interface is needed for requests from monitor terminals.

Hence three interface types are needed:

1. An interface for X.500 DAP requests that arrive from other functional elements. These are the main source of requests for a SDF. The requests may be both reads and updates. Even if the SMF model is based on TMN and X.700 CMIP, the SDF database is updated via the X.500 interface. However,

if the SDF element is managed, the management functions will probably use X.700 CMIP. Hence the SDF has two update interfaces depending on the nature of the update. Data is managed via X.500, database functionality (entity-related data) is managed via X.700.

2. An interface for now obsolete CS-1 INAP requests that must be supported for backward compatibility. These requests should be relatively rare since all implementations should use DAP. Nevertheless it must be supported.
3. An interface for high-level query language requests that arrive from monitoring staff. These requests are database management requests that arrive directly from a terminal. This interface is mandatory for management reasons. A SMF is not intended for this kind of database management, although it can be used as such if the SMF is embedded to the same physical element with the SDF.

The listed three interface types are sufficient for CS-2. However, in the future also an interface for Object Distributed Processing (ODP) channel is needed if long term IN architecture plan is to be compatible with TINA. TINA architecture is based on the ODP models and interfaces.

The SMF interfaces are more problematic since they have not been defined in CS-2. It is difficult to exactly state which kind of interfaces are needed for an appropriate SMF database architecture, but TMN interfaces can be given as the first assumption.

The TMN interfaces are based on OSI X.700 Common Management Information Protocol (CMIP). As such, the SMF data management system must support two types of interfaces:

1. TMN management requests that arrive from Management Information System (MIS) Agents. A MIS agent is needed to connect a managed element to a TMN manager.
2. TMN notifications that arrive from managed objects. A managed object can notify its manager that something special has happened. The manager is responsible of responding the request. If TMN notifications are added to the SMF data architecture, the resulting database is an active database. However, this is not a mandatory requirement since the manager can accept notifications and translate them to appropriate database requests. That way the database has to answer only regular TMN requests.
3. High-level query language requests. These are similar to SDF requests.

3.2 Logical data model

The CS-1 and CS-2 recommendations do not specify anything about SDF or SMF database models. In CS-2 it is only stated that SDF may be based on a database management system (ITU-T 1996). Hence no requirements raise from the recommendations. This is a good design policy since it allows the use of different types of database management systems without breaking recommendations. But on the other hand, it does not give much to the system designer.

Some requirements can be derived from the interfaces and from the possible IN long term architecture plan that intends to join TMN and IN under TINA. From these sources we can derive the following candidates for a SDF and SMF logical data model.

ITU-T X.500 model

This is a logical choice for the SDF data model since the interface of SDF is based on X.500. The drawback of it is that it is intended for a global directory model and hence is not necessarily sufficient for a logical data model. The model is somewhat old-fashioned since it is related to the hierarchical model that has not been used in decades.

ITU-T X.700 model

If SMF is based on TMN, X.700 model is a logical choice for the SMF logical data model. The model is intended for data management, so it can be used in a database management system. It is also designed after the X.500 data model which makes it a more developed one. The drawback of the model is its lack of acceptance in database fields. No general commercial databases are based on X.700 data model, although X.700 implementations do exist.

OMG-based model

Once the long-term IN architecture plan comes true, the IN distributed functional plane architecture can be based on TINA object modelling, which in turn is based on Object Management Group (OMG) core object model (OMG 1992). Hence an OMG-based model is a good candidate for both the SDF and the SMF database architecture. In the short term, an OMG-based model is not as attractive as the ITU-T models. We have shown that not all X.500 and X.700 features can be mapped directly to an OMG-based model (Taina 1994). A special layer is needed to implement X.500 and X.700 special features. In the long term, however, this approach is the best one. It allows the use of OMG Object Request Broker (ORB) architectures which is a necessity in a TINA platform.

ODMG-93 model

One interesting candidate for the logical data model is ODMG-93 (Cattel 1994). It is an OMG-compliant data model that is intended for object-oriented databases. The advantage of the model is that several database vendors are going to publish ODMG-93 based object database management systems. It can also be used directly with ORB architecture. Both OMG and ODMG models are widely accepted and research on both models is active.

Choice of model

The chosen model for a database management system can be any of the three models, or it can be something completely different. Currently most database management systems are based on the relational model, which works well in IN database architectures. However, since IN is already mostly object-oriented, and TINA is completely object-oriented, an object-oriented database management system simplifies IN data modelling.

3.3 Core database issues

In the core database issues we deal with the requirements that the IN environment sets to the database architectures and implementations. These requirements raise both from the ITU-T recommendations and from the general telecommunications recommendations.

We consider the requirements from five points of view: 1) request throughput and response time that deals with the issue of how much user requests must be served in a specific time, 2) fault-tolerance that deals with data availability issues, 3) data security that deals with database security issues, 4) data distribution issues that deals with requirements for distributed data management, and 5) relaxing ACID-properties (Atomicity, Consistency, Isolation, and Durability) that deals with database transaction model issues.

Throughput and response time.

The recommendations of CS-1 and CS-2 state that SDF must offer real-time access to data (ITU-T 1993a, 1996). It does not state what the exact real-time requirements are. We assume that real-time in this context does not imply a real-time database, e.g. a database with timing constraints. Instead, the requirements imply a high throughput database.

Nevertheless, exact response times in SDF are often more important than throughput. This is due to the fact that most SDF read requests are for SCF or SRF logic programs that have exact time limits. If the database cannot give a response within a specific time limit, it is better not to waste resources and hence abort the request. As a result of this, SDF request management policy should favour predictable response times with the cost of less throughput. The best alternative is that the database can guarantee that all requests are replied within a specific time interval. Unfortunately this is not possible unless the time interval is very high or the database workload is very low. Otherwise a single overloaded condition ruins the guaranteed policy.

The average time limit for a read request in SDF is around 50 ms. About 90% of all read requests must be served in that time. For updates, the time limits are not as strict. It is better to finish an update even at a later time than abort the request.

In SMF, the roles of throughput and response times are reversed. Since the primary task of SMF is management and management operations should be committed, it is more important to finish as many management requests as possible than finish less requests with predictable times. SMF read requests are either high-level query language commands or requests that are related to management updates. They are more liberal with time limits than SDF reads because the result of the request is valid even after a time period.

The only SMF request that may need predictable time limits is a TMN notification request. Since the notification describes an unusual condition in the managed entity, it must be processed as soon as possible. A regular request processing policy may not be efficient enough for the notification. Instead, the notification can be served better if the request has a specific time limit. In this case it should also be guaranteed that the notification is be served in time.

Fault-Tolerance

The IN functionality is dependent on high availability of SDF data. The total non-access time of data should not exceed seconds in a year and it is allowed to be only fractions of a second at a time. Two things should be noted in case of a system failure: 1) the effect of it for the call (call is dropped), and 2) the effect of it for charging (user calls for free). The first can be acceptable as long as the probability of it is low. The latter is not acceptable. Hence SDF fault-tolerance should be tailored for continuous data access, especially in charging situations.

Also SMF database should be fault-tolerant but since it is for service and network element management, it is not as necessary to service execution than SDF data management. We think that the fault-tolerance level can be relaxed in management. The database should be fault-tolerant, but the allowed down time can be minutes in a year instead of seconds, and the recovery time can be longer than in SDF data management as long as data is available. It is more important to ensure that committed data in SMF will not disappear in a failure than support full fault-tolerance.

Data Security

Security is one of the major issues in CS-2. A SDF must support secure requests, such as PIN identification. Also requests from external networks must be secure. It is not clear from the recommendations how high a security level is needed for the IN database architecture. It is probable that a well-known Bell-LaPadula security model (Bell *et al.* 1976) is sufficient. In the model, a request is allowed to read-access an object only if its clearance is the same or higher than the object's classification. A request is allowed to write-access an object only if its clearance is the same or lower than object's classification. In other words, requests are allowed read data that is less or equal secure than their own security level, and are allowed to update data that is less or equal secure than their own security level.

The SMF security level does not differ from the SDF security level. Although management security issues are not addressed in CS-2, the SMF may receive requests from external networks or external users. The Bell-LaPadula model should be sufficient for the SMF security model.

A simple and often used addition to the Bell-LaPadula security model is to define private security groups. A member of a private security group may access all data that is in her security group. For other data, the Bell-LaPadula model used. This addition allows an user to have private data that only she can access. If user data is at a higher security level than user access level, the users cannot read any other than their own private data. They can still write other data if they are allowed to have a reference to it.

Data distribution and replication

As the name states, IN Distributed Functional Plane is distributed. This definition encloses both the fact that IN DFP has several functional elements, and the fact that the plane can have several instances of the same functional element.

Although the IN DFP defines a distributed model, it does not force the architectures to be distributed. In principle a monolithic SDF could serve all IN data requests in a small network. Such a solution is not practical since the database can easily become a bottleneck. The approach can be used if the workload can be analysed and the network is reliable enough.

If the IN service database architecture is distributed, a logical distribution level is to define each SDF as a node in a distributed database. The CS-2 model supports this approach well since a SDF can interact with other SDFs in the same or in an external network. The X.500 data model supports distribution since it is a distributed directory standard. The recommendations state that SDF architecture must support location transparency (ITU-T 1996).

In SMF, distribution is even a more natural feature than in SDF. The TMN model is naturally hierarchical and distributed. If SMF follows TMN, the SMF model is also distributed. Also most management requests are distributed because they can update several network elements which can be geographically far from each other. Although this can be handled with a single management node, the management system is usually distributed such that it is closer to the elements that are managed.

If SDF and SMF are distributed, also data replication becomes an issue. In principle distribution does not require replication. In practice it does, since otherwise often referenced data has to be requested from an external source. It is better to replicate data closer to the location of the requester. This is especially true in SDF, since most SDF requests are read-requests. If updates dominate, like probably in SMF, the advantages of data replication are lost since all updates must be done to all replicas in the distributed database.

Relaxing ACID-properties

Finally, the ACID-properties summarise how a database management system processes requests. A requests spawns one or more transactions into one or more database nodes in a distributed database. If the database is not distributed, a request spawns one transaction. When a transaction is finished, it commits the changes it has made to the database. Usually only then the changes are visible to other transactions. A transaction has ACID-properties (Atomicity, Consistency, Isolation, and Durability) that describe how it can coexist with other transactions. Some of the properties can be relaxed in order to gain better throughput or response time.

- **Relaxing Atomicity**. Transactions are atomic. The updates a transaction makes to the database are either accepted completely or not accepted at all. There is no reason to relax atomicity in IN databases. The same result can be gained by allowing requests to spawn several transactions all of which are atomic.
- **Relaxing Consistency**. A transaction moves the database from one consistent state to another. If the database is consistent before the transaction execution, it will be consistent after the execution. Consistency should not be relaxed either. If it is relaxed, the database becomes inconsistent in which case the data no longer reflects the real-world situation. The inconsistent database can no longer return right answers.
- **Relaxing Isolation**. In strict isolation transactions should not make their updates visible to the database before commits, and thus transactions do not interfere with each other. This can be relaxed in SDF databases but not in SMF databases. The result of the relaxation is that a transaction may sometimes read wrong data in order to finish in time. This may or may not be tolerable, depending on the type of the transaction. In management it should never be tolerable since read data affects management updates. Reading uncommitted data can lead the database into an inconsistent state.
- **Relaxing Durability**. Once a transaction is committed the changes made in it will not be lost because of subsequent failures. This can be relaxed in SDF databases to certain extent. In busy conditions it is acceptable that committed writes may be lost, as long as the probability of loosing a committed write is small and it goes to zero in time. It should not be relaxed in management databases. If for instance billing information is lost due to relaxed durability, the result of the data loss can become very costly to the charging operator.

We have summarised both the SDF and SMF database manager requirements in Table 1. The requirements for the database managers are mostly similar. If a compromise can be defined in object models, throughput versus response times, and ACID-properties, it is possible to design a database management system that can be used in both SDF and SMF. Since the requirements for data access are stricter in SMF than in SDF (SMF needs full ACID-properties), the combined

SDF/SMF architecture may become too inefficient for IN. On the other hand, since SDF databases should offer estimated response times, long SMF transactions may suffer from update aborts and lack of resources since short high-priority SDF transactions may steal all resources from them.

Table 1 Summary of SDF and SMF database requirements

Requirement type	*SDF*	*SMF*
Interfaces	Multiple application interfaces	Multiple application interfaces
Data model	Object-oriented logical data model	Object-oriented logical data model
Response times vs. Throughput	Predictable response times in favour of throughput	Throughput in favour of predictable response times
Data security	Full data security	Full data security
Fault tolerance	Full fault-tolerance	Relaxed fault-tolerance
Distribution	Data distribution and replication	Data distribution and replication
ACID-properties	Full Atomicity and Consistency, relaxed Isolation and Durability	Full Atomicity and Consistency, Isolation, and Durability

4 DISCUSSION AND CONCLUSIONS

In this paper we have summarised the requirements that an IN database architecture must fulfil both in service execution (SDF) and service management (SMF that uses TMN). The requirements are all derived from user needs. The SDF users are IN DFP functional entities such as SCFs, or external applications from other telecommunication networks that want to access data either via an INAP or X.500 DAP interface. The SMF users are management agents that want to access X.700 management data. If X.500 and X.700 requests access different data, the database management system may support both X.500 and X.700. However, when the requests have a common subset of data the database management system must have a mapping between the two models, plus a mapping to the actual database data model. The actual database model must then be complete enough to support both models and also database-related aspects such as transactions and data integrity. This can cause problems since the models are not 100% compatible. If the mapping becomes too complex, either SDF and SMF data must be stored separately or one of the models must be simplified.

Similarly, when SDF and SMF access common data in a database, the created transactions may conflict with each other. SDF transactions are usually short high priority transactions. SMF management transactions are complex and long, and often they cannot have as high a priority as SDF transactions. In a database that favours predictable execution times this can lead to a situation where SMF transactions cannot meet their deadlines or have to wait forever because higher priority SDF transactions steal the needed resources.

A possible way to implement different database interfaces is to build a set of interpreter processes each of which handles a certain request model. In order to do this an uniform interpreter-database connection is a necessity. The interpreter may either use an embedded database language or a special interpreter language. Both alternatives are equally good and possible to implement. In fact a good interpreter request language is very close to an embedded database language.

As a conclusion, it should be possible to build a database architecture that fulfils the listed requirements as long as the restrictions above are taken into account. It might not be possible to build a database architecture that can best answer both SDF and SMF requests. If the model differences between service use and management may be handled with an efficient model mapping then it should be possible to build a database architecture that can support both SDF service transactions and SMF management transactions. Such a database management system must guarantee that the SMF management transactions have a chance to finish execution in time.

6 REFERENCES

Bell, D.E. and LaPadula, L.J. (1976) *Secure computer systems: Unified exposition and multics interpretation*. Technical report, The Mitre Corporation.

Cattel, R.G.G. (1994) *Object Database Standard: ODMG-93 Release 1.1*. Morgan Kaufman Publishers, San Francisco.

Chatras, B. and Gallant, F. (1994) Protocols for Remote Data Management in Intelligent Networks CS1, in *Intelligent Network'94 Workshop Record*, IEEE Communications Society.

Demounem, L. and Zuidweg, H. (1995) On the Coexistence of IN and TINA, in *Proceedings of TINA'95 conference*. TINA-C, Red Bank, N.J.

Eurescom (1993) *Fact-finding Study on Requirements on Databases for Telecom Services*. Technical report, Eurescom, Heidelberg, Germany.

Eurescom (1994) *TMN Management of Pan-European IN Based Freephone Service*. Technical report, Eurescom, Heidelberg, Germany.

ITU-T Recommendation Q.1214 (1993a) *Distributed Functional Plane for Intelligent Network CS-1*. ITU, Geneva, Switzerland.

ITU-T Recommendation Q.1218 (1993b) *Interface Recommendation for Intelligent Network CS1*. ITU, Geneva, Switzerland.

ITU-T Draft Recommendation Q.1224 (1996) *Distributed Functional Plane for Intelligent Network CS-2*. ITU, Geneva, Switzerland.

Kerboul, R., Pageot, J.-M. and Robin, V. (1993) Database Requirements for Intelligent Network: How to Customize Mechanisms to Implement Policies, in *Proceedings of the 4th TINA Workshop*. TINA-C, Red Bank, N.J.

Khoshafian, S. and Baker, A.B. (1996) *MultiMedia and Imaging Databases*. Morgan Kaufmann, San Francisco.

OMG (1992) *Object Management Architecture Guide* - Revision 2.0 - Second Edition.

Raatikainen, K.E.E. (1993) A Framework for Evaluating the Performance of IN Services, in *Proceedings of Workshop on Intelligent Networks*. Lappeenranta University of Technology, Finland.

Raatikainen, K.E.E. (1994) *Information Aspects of Services and Service Features in Intelligent Network Capability Set 1*. Technical report, University of Helsinki, Department of Computer Science, Finland.

Taina, J. (1994) *Evaluation of OMG, ODMG, X.500, and X.700 Data Models*. Technical report, University of Helsinki, Department of Computer Science, Finland.

7 BIOGRAPHY

Juha Taina received the MSc degree in computer science from the University of Helsinki, in 1992. He is currently a Ph.D. student in computer science at the Univeristy of Helsinki. His research interests include real-time databases, object-oriented real-time database models, and databases in telecommunications.

Kimmo Raatikainen received the Ph.D. degree in computer science from the University of Helsinki, in 1990. He is currently an associate professor in computer science at the University of Helsinki. He is a member of ACM, IEEE (Communications and Computer Societies), and IFIP TC6 Special Interest Group of Intelligent Networks. His research interests include nomadic computing, telecommunications software architectures, and real-time databases.