University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1995-17

Requirements Analysis for Database Services in Telecommunications*

Juha Taina

Helsinki, June 1995

Report C-1995-17

University of Helsinki
Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, FINLAND

* This report is based on work done in Darfin-project (Project No 30018) funded by Telecom Finland.

Requirements Analysis for Database Services in Telecommunications†

Juha Taina

Report C-1995-17
17 pages
Helsinki, June 1995

Abstract. This report summarizes the requirements identified for a Telecommunications database use in project Darfin (Database ARchitecture for Intelligent Networks). The requirements described here are for the final and complete Darfin TDMS (Darfin Telecommunications Data Management System) architecture. This report is not a documentation of the Darfin OODBMS prototype that is under construction in project Darfin.

Darfin TDMS is a distributed object oriented data management architecture for Intelligent Networks (IN) and Telecommunications Management Networks (TMN). It is intented to store and maintain all persistent data that is needed in the two networks. Distributed Darfin OODBMS architecture may be divided into four layers: User layer, Interpreter Layer, Database Layer, and Database Kernel Layer. The User Layer defines the external world outside Distributed Darfin OODBMS. The Interpreter Layer defines the interfaces from the User Layer to the actual database. The Database and Database Kernel Layers define Darfin OODBMS (Object Oriented Database Management System) behavior that is responsible of actually maintaining the data stores needed. In this report we summarize the functional requirements that Distributed Darfin OODBMS and Darfin OODBMS must fulfil in IN and TMN. The requirements vary between IN and TMN due to the different nature of the networks.

Key Words: object oriented database architecture, real time database requirements, intelligent networks, telecommunications mangement networks

CR Classification: C.2.4, C.3, C.4, H.2.1, H.2.4

†This report is based on work done for Darfin-project (Project No 30018) funded by Telecom Finland.

Contents

1. Introduction **1**

2. General **2**

3. User Layer Requirements **4**

3.1 Interface requirements 4

3.2 Request requirements 6

4. Interpreter Layer requirements **6**

5. Database Layer and Database Kernel Layer requirements **8**

5.1 Data distribution and replication 8

5.2 Database aspects 9

5.3 Transactions and concurrency control 11

5.4 Fault tolerance and recovery 13

5.5 Object operations 14

5.6 Other database and database kernel view requirements 14

6. Other requirements **15**

6.1 Environment 15

7. Summary **15**

References **17**

1. Introduction

In project Darfin (Database ARchitecture for Intelligent Networks) we have worked on a suitable database architecture that can fulfil the telecommunications requirements. This report summarizes the requirements we have identified in our work. It also defines the final product requirements for Darfin TDMS (Telecommunications Data Management System).

Currently we are working on the first prototype version of Darfin TDMS with a strongly reduced set of requirements. This report does not refer to the prototype but the final Darfin TDMS product.

Darfin TDMS is a data management architecture for open Intelligent Networks (IN) and open Telecommunications Management Networks (TMN). It consists of a set of of interfaces to the stored and maintained data and an object oriented database management system (OODBMS) to maintain data. The actual OODBMS is called Darfin OODBMS. It is intented to store persistent data that is needed in IN and TMN networks. The two networks have different characteristics and database views to data.

The data management requirements in IN are different from the data management requirements in TMN. In IN, the data management system is mostly used for storing data that is needed in a single call, such as user profiles or number translation. In TMN the data management system is used for storing information about the TMN network elements that the TMN manages, or for the network maintenance data. The differences affect the stored data in Darfin OODBMS, and possibly Darfin OODBMS architecture itself.

This report summarizes the functional requirements that are needed for the Intelligent Network and Telecommunications Management Network data management views. We use terms IN-view and TMN-view when we refer to the Intelligent Network database and to the Telecommunications Management Network database, respectively. Both IN-view and TMN-view define a view to the Darfin OODBMS managed data. The architectures of IN-view and TMN-view are not necessarily the same.

Usually IN-view is used from a Service Control Function (SCF) or from another single IN source. It is usually accessed quite locally. Darfin OODBMS is closely related to a Service Data Function (SDF) in a local node, such as a Service Data Point (SDP). There may be other SDF:s in the network. It is possible to access them from the calling function or from the data function. In such cases Darfin OODBMS in IN-view is closely related to a database node in a distributed database. A similar concept is true in TMN-view. A single Darfin OODBMS in TMN-view is usually accessed locally and there may be other accessible Darfin OODBMS TMN-view nodes in the network. The total set of TMN nodes or SDF:s together is the whole distributed data management system. Usually data distribution is invisible to the users. From the user's point of view both IN-view and TMN-view are data stores for persistent data that are needed in the networks. When it is important to distinguish the total set of database nodes from a single node we use term Distributed Darfin OODBMS. Usually the requirements for both Distributed Darfin OODBMS and Darfin OODBMS are alike.

This report is organized as follows. Section 2 is about the different layers that can be identified from Darfin TDMS. The definitions listed in the Section are used in the rest of the report. Section 3 defines the requirements that affect user interfaces to Darfin TDMS both in IN-view and TMN-view.

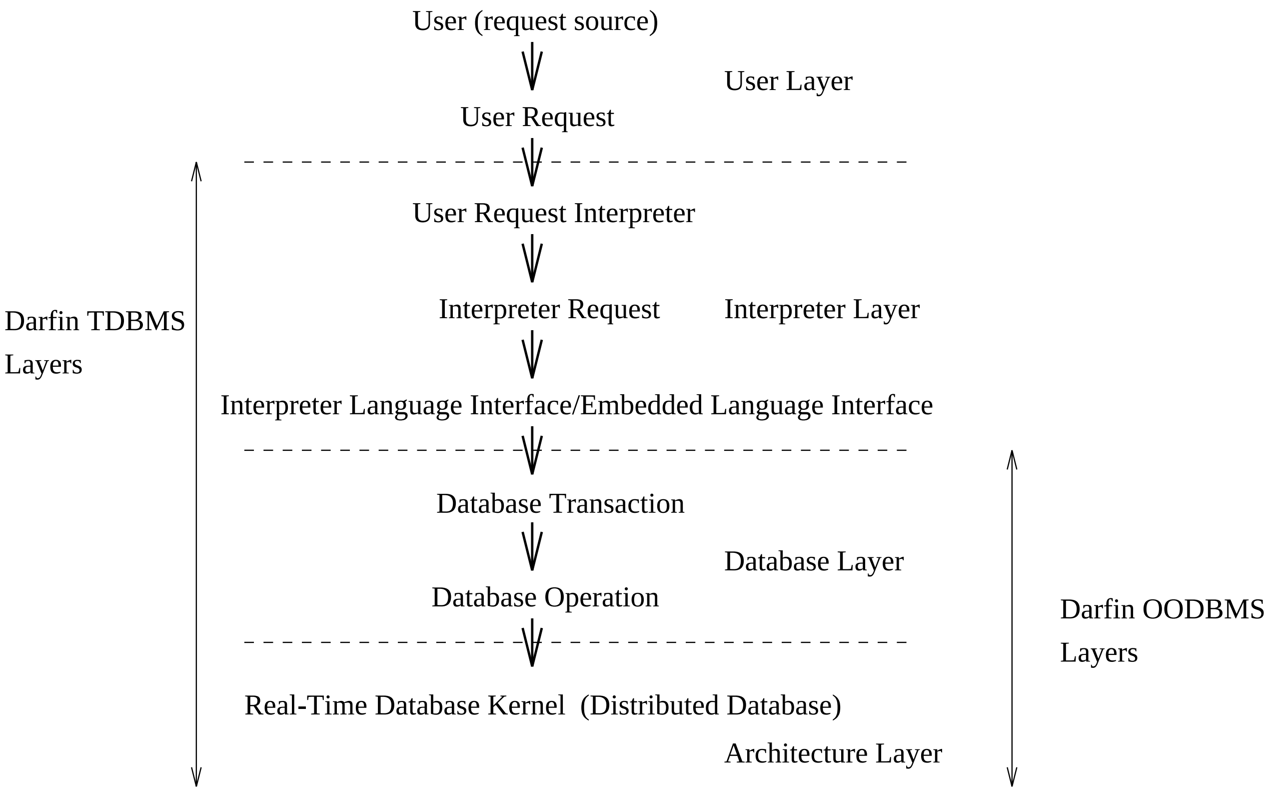

Figure 1: Hierarchical views to Darfin database

The Section defines the possible user interfaces and the request types that the interfaces must fulfil. Section 4 defines the requirements that affect the interpreter level between users and the database. The Section defines the requirements that allow users to use several different interfaces in Darfin TDMS to access the actual Darfin OODBMS both in IN-view and TMN-view. Section 5 defines the requirements that Darfin OODBMS must fulfil in processing both IN-view requests and TMN-view requests. Section 6 defines other requirements to the Darfin TDMS that cannot be classified by the previous Sections or that affect more than one of the other Sections. This includes environment and software requirements. Section 7 is the summary.

2. General

Darfin TDMS terminology may be ordered into a hierarchical structure that is divided into four layers: User Layer, Interpreter Layer, Database Layer, and Architecture Layer (figure 1).

The top layer is the User Layer. It defines the interface that may be seen from the world outside. The User Layer defines the terms User and User Request.

- *A User* is any external source that is allowed to use the Darfin TDMS. A User is usually a network element, such as a Service Control Function in IN. Thus a User is a request source to Darfin TDMS. All Darfin OODBMS and Distributed Darfin OODBMS functional requirements are based on the requirements from the Users.

- *A User Request* is a request that a User sends to the data management system. The User Requests may be of different types. The structure of a User Request is not limited to any protocol in this layer.

The second layer is the Interpreter Layer. It defines the interface between a user and the Darfin OODBMS database. This Layer allows the database architecture to maintain several different request interfaces to the users. The Interpreter Layer is used to translate the different types of user requests to the database in a uniform manner. It defines terms User Request Interpreter, Interpreter Request, Interpreter Language Interface, and Embedded Language Interface.

- *A User Request Interpreter* is a module in Darfin TDMS architecture that is outside the actual Darfin OODBMS. It interprets the User Requests that it understands and that are addressed to it. It is possible to have several different User Requests Interpreters for different User Request types, such as X.700 and X.500.
- *An Interpreter Request* is an interpreted User Request to the Darfin OODBMS. The User Request Interpreters may use Interpreted Requests to access the Darfin database.
- *An Interpreter Language Interface* is a language interface from Interpreters to Darfin OODBMS. It consists of interpreter operations that together form the interpreter request sent to Darfin OODBMS. Typical interpreter operations are Read, Write, Delete, Begin Transaction, Commit, and Abort. Alternatively the Interpreter Language may be a high level language, such as Object Query Language (OQL).
- *An Embedded Language Interface* is a language interface from Interpreters to Darfin OODBMS. It is a high level programming language that has embedded database operations such as a C++ binding in an object oriented database management systems. A User Request Interpreter may use an Embedded Language Interface to access Darfin OODBMS database.

The next layer is the Database Layer. It is the first Darfin OODBMS layer. Its terminology defines the logical operations in Darfin OODBMS. In the Database Layer it is not important who has sent what requests as long as the Darfin OODBMS is capable of understanding the interpreted actions. The Database Layer defines terms Database Transaction and Database Operation.

- *A Database Transaction* is an undivisible unit of operations in Darfin OODBMS database that are either all accepted or none accepted. A database moves from one consistent state before the transaction to another consistent state after the transaction. An Interpreter Request usually creates a single transaction that may be further divided into several subtransactions of arbitrary dimension.
- *A Database Operation* is any operation that touches the database such as a Read, a Modify, or a Delete. All operations are either encapsulated into transactions with begin-transaction and end-transaction commands in the Interpreter Requests or implicitly cause encapsulation, for instance in case of a single read operation.

Finally, the last layer is Architecture Layer. It defines terms that affect different architectures. The previous levels were independent of the architecture definitions. The Architecture Layer defines terms Real-Time Database and Distributed Database.

- *A Real Time Database* is a database where transactions have deadlines. The database has at least three different types of transactions:
  - *Soft Transactions* that should be finished before the deadline but that are still valuable if the deadline is exceeded.

  - *Firm Transactions* that should be finished before the deadline. If a firm transaction cannot be finished in time the result of the transaction is useless and the transaction is discarded.
  - *Hard Transactions* that must be finished before the deadline. The failure of finishing a hard transaction in time may cause severe damage. We hope that such transactions do not exist in Telecommunications databases.
- *A Distributed Database* is a database where information is stored into several distributed database nodes. In a regular database all information is stored into a single node. The dependence level between database nodes in a distributed database varies in different systems.

3. User Layer Requirements

From the User's point of view Darfin TDMS is a store of persistent objects that are organized in a way that is compatible to the User. For instance, if the User is a Service Control Point using X.500 User Requests, it uses the IN-view and sees the Darfin OODBMS database as a store of X.500 nodes. If the User is a TMN agent using X.700, it uses the TMN-view and sees the database as a store of X.700 objects. The User does not have to know about the actual Darfin OODBMS data model, the internal structures needed to maintain the database, or data distribution to the distributed database nodes. However, in some cases it is desirable that the User is allowed to see all the details in the system.

Darfin TDMS must answer the interface requirements both of possible Users and of possible database uses in IN-view and TMN-view. The requirements are listed in the following two subsections.

3.1 Interface requirements

Darfin TDMS interfaces are dependent on the requests and request sources. Darfin TDMS must be able to communicate at least with IN Service Control Functions (SCF), with other Service Data Functions (SDF), Service Management Functions (SMF), access functions from other networks, and maintenance functions that access the database node directly from console or other without using the management network.

Both IN-view and TMN-view must support distributed requests. The data distribution may be visible or invisible to the User Layer. In Darfin OODBMS both IN-view and TMN-view must support at least X.860, X/Open, and Tuxedo standards for distributed transactions between database nodes or heterogeneous databases.

We have identified the following request sources, their views to the database, and the default actions:

- Requests from service control functions (SCF $\rightarrow$ SDF). It is probable that the next ITU-T recommendation for database access in IN will define the interface to be a subset of OSI X.500 directory recommendation. The request sender is a directory user agent (DUA). A directory system agent (DSA) is needed to translate such requests. Thus this is a DUA $\rightarrow$ DSA request. The database is seen from IN-view. Data distribution to several nodes is visible.

- Requests from other service data functions (SDF $\rightarrow$ SDF). This is a similar request than the previous SCF-based request. Data distribution is visible as the request can be considered to come from another database node inside the distributed database. This is a DSA $\rightarrow$ DSA request. The database view may either be the IN-view or TMN-view.
- Requests from external networks. The requests see a logical X.500 view to the database. Distribution is not visible. Communication with the database goes through an interface that hides the distribution. The hiding interface offers a DSA-connection to the request source from the external network and a DSA $\rightarrow$ DSA connection to the distributed nodes. The database view is IN-view.
- CS-1 INAP requests. The ITU-T recommendations for intelligent networks define CS-1 INAP commands. The commands must be interpreted to the database. This is a SCF $\rightarrow$ SDF connection. Data distribution may or may not be visible. In the latter case the requests are sent to the distribution hiding interface. The database view is IN-view.
- IN management requests. As of this writing the IN service and customer management is not standardized. The management is left to Service Management Functions (SMF). The probable interface to the management functions is X.700. This is a SMF $\rightarrow$ SDF connection. Data distribution may or may not be visible. The database view is TMN-view, although the managed data is in IN-view.
- TMN management requests. Currently TMN uses X.700 in management functions between Management Information System (MIS)Agents. This is a MIS manager $\rightarrow$ MIS agent connection. A Management Information Service agent (MIS) is needed to handle the connection to TMN. Data distribution may or may not be visible. In the latter case the requests are sent to the distribution hiding interface. The database view is TMN-view.
- ITU-T Open Distributed Processing (ODP) channel. This request type is yet undefined. The database is either in TMN-view or in IN-view.
- High level query language requests. These are usually high level queries where the source is an end user or a software that has embedded high level query commands. The request is sent to exactly one node and it is usually sent from console. Distribution is not visible nor are other nodes. The queries must be interpreted to the database. The view to the database is TMN-view.

The most common request sources in the previous list are based on X.500 directory. This causes problems in both IN-view and the TMN-view. For instance, in IN-view the logical X.500 structure is customer-oriented. A teleoperator has a number of customers that have services. In TMN-view the logical structure is service-oriented. A teleoperator has a set of services that have customers. It is possible to create an uniform view in X.500 directory by using alias-entries but this might be inefficient.

The Darfin OODBMS database should be efficient enough for IN-view that has tight response time limits. A pure X.500 directory approach in data organization may not be good enough. For such situations the database must be able to offer a fast connection to data for local request sources that know what they are doing. This means an ability to see and use the Darfin OODBMS core, and thus bypass interpreters. The internal data structure may be based on any efficient solutions as long as such solutions may be used by local request sources. This requirement adds a new request source:

- Direct requests to the database. Database interface functions are accessed directly. Distribution is visible. The view to the database is either IN-view or TMN-view.

Such a request is comparable to an interpreted high level query. It is desirable that a direct high-level query language request is allowed to see the Darfin OODBMS core.

3.2 Request requirements

There are five basic types of requests needed in IN-view services and service features of Capability Set 1 [Raat94b].

1) Retrieval of structured objects from persistent subscriber data objects. Some of the retrievals trigger a later update. For example, if Freephone service provides call routing statistics to the subscriber, an update is necessary.
2) User management actions that modify persistent subscriber data objects. These actions must be protected against unauthorized use. Some updates may also have to be verified against other criteria. For example, the owner of a telephone number must have the possibility to cancel or prohibit call forwarding to his or her number.
3) Verification of Personal Identification Number (PIN). For security reasons the PIN verification must be done in the database system when services are globalized. The actual PIN cannot be delivered into a foreign network.
4) Writing sequential log records (see serialization in IN-view in Section 2).
5) Mass calling and Televoting. They need high-volume small updates that must be serialized (see serialization in IN-view in Section 2). A special aspect of these updates is that they can be done in large blocks.

It is not possible to form a corresponding list of basic types of request to TMN-view. In a general level TMN-view must answer two kinds of requests:

1) Network management requests. These requests cause highly distributed operations. Every managed network element has its own management agent that uses a local TMN-view database node. Such nodes are very small in size. A common network management request affects several such elements and thus causes a distributed database operation.
2) Telecommunications service and customer management requests. These requests are less distributed in nature than the previous type requests, although a service update might affect several nodes. However, the TMN-view to service and customer maintenance is closer to the IN-view database. The database nodes are usually large in size.

In a detailed level TMN-view must answer X.700-based requests that reference objects and object operations.

4. Interpreter Layer requirements

The Interpreter Layer is the interface layer between Users and Darfin OODBMS. It is responsible of interpreting User Requests that are sent to Darfin TDMS to Darfin OODBMS. The requirement from User View to Interpreter View is to hide the internal structure of the Darfin OODBMS database and the same time offer multiple interfaces to the database.

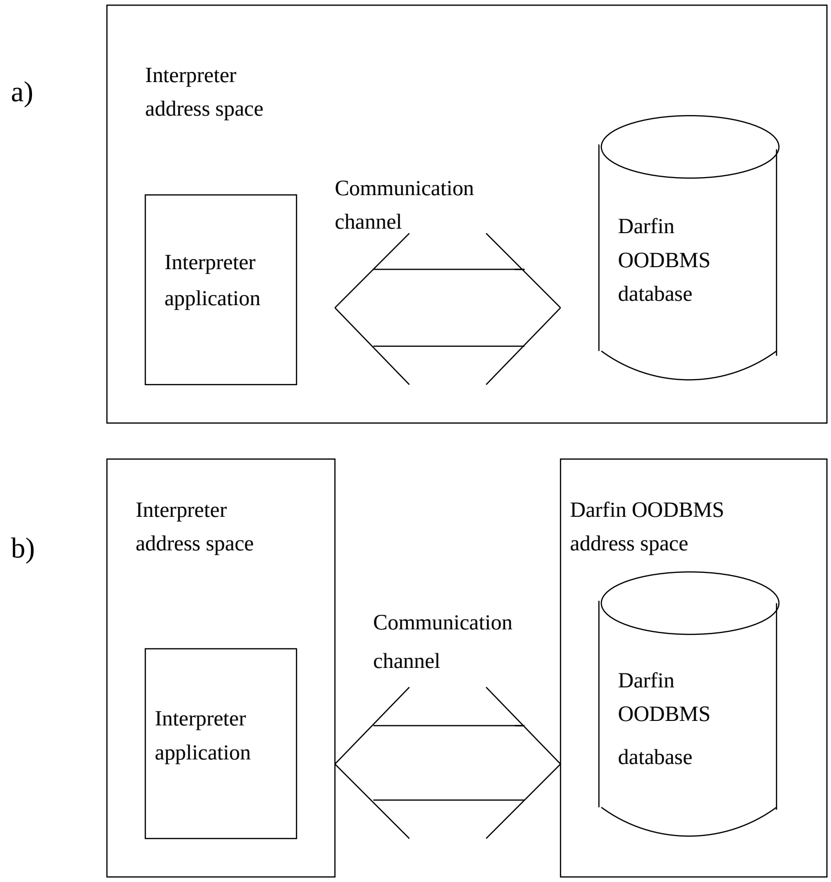

Figure 2: Interpreter access to Darfin OODBMS. a) Direct access to the database using a common address space and a language binding b) Indirect access to the database using different address spaces and an Interpreter Request Language

A User Request Interpreter is responsible on interpreting any User Request that is addressed to it. Darfin TDMS may have several different interpreters in it, and it is possible that some interpreters work only on certain nodes in Distributed Darfin OODBMS.

It is possible to expand the Darfin OODBMS with new User Request Interpreters just like it is possible to have new programming language interpreters in an operating system. Each User Request Interpreter works independently of the others.

The result of a User Request sent to the appropriate User Request Interpreter is an Interpreter Request. There are two ways to create the interpreter request (figure 2).

1) *Using an Embedded Database Language.* An Interpreter is written in a high level language that supports database operations. For instance, most object oriented database management systems offer a C++ binding for accessing the database. An Interpreter Request consists of database language sentences. Both the Interpreter and the Database objects share the same address space, and it is possible to access any object in the database within the access limits (figure 2 a)).

2) *Using an Interpreter Request Language.* An Interpreter Request consists of Interpreter Operations that together form the Interpreter Request language. The interpreted request is encapsulated into a transaction and sent to the database. The address spaces of the Interpreter and the database are different (figure 2 b)). The language must be flexible enough to fulfil the following requirements:
   - It is possible to interpret any legal User Request into a legal Interpreter Request that has the same functionality, either in IN-view or in TMN-view.
   - A single Interpreted Request may contain several transactions and subtransactions to the Darfin OODBMS.
   - Transactions, subtransactions, and operations may have deadlines.
   - All objects, classes, and operations are accessible from the language.
   - All low-level data structures and operations are accessible from the language.
   - The language supports conditional and repeated operations.
   - The language supports distributed operations.

The final decision between an embedded database language and an interpreter request language is yet undefined. The former is more flexible, the latter gives more choices to Darfin OODBMS core to process transactions.

5. Database Layer and Database Kernel Layer requirements

Database Layer and Database Kernel Layer requirements define the core functionality of Darfin OODBMS. The requirements rise from the User Layer but the implementation requirements are defined here.

The most important requirement for Darfin OODBMS is expandability. Darfin OODBMS is an open modular system. It is possible to revise, modify and expand the database software to for new and revised requirements. The interfaces between different modules are clear and without side effects. It is possible to replace a module with another module that performs the same task.

Both IN-view and TMN-view are object-oriented implying that stored data is an abstraction of real world objects. This differs from value-oriented views where stored data is an abstraction of information tables.

The database is a store for persistent objects that can be accessed from both IN-view and TMN-view. It is possible to dynamically store both new objects and new object classes to Darfin OODBMS. All low-level operations to the database are encapsulated into transactions.

5.1 Data distribution and replication

Both IN-view and TMN-view must offer interfaces to various request types. While the internal structure of the database is not important to the database interfaces, choosing between a distributed database and a one-node regular database affects the interfaces. We have chosen a distributed database architecture for the following reasons.

1) Most requests in both IN-view and TMN-view are geographically distributed. It is easier to process the requests in a local node within a distributed database than with a global single node that is geographically far from the request source.

2) The database access requirements are hard to fulfil with a single-node database. As most requests to the database are relatively local in nature it should be relatively simple to distribute maintained data to gain an acceptable load into the distributed database nodes.
3) A node in a distributed IN database is a database equivalent of a Service Data Function (SDF) in IN. This allows us to use terms service data function (SDF) and distributed database node interchangeably.
4) It is possible to build a logical global interface into a distributed IN or TMN database and thus hide the data distribution.

In Distributed Darfin OODBMS, all Darfin OODBMS nodes are independent and are themselves regular databases. The set of all database nodes is called the Distributed Darfin OODBMS.

Darfin OODBMS must support kernel-level distributed operations between database nodes. This implies either a global operation that affects the global view to data, both in IN-view and TMN-view, and a distributed operation between several nodes or heterogeneous databases. The exact protocols visible to the users are listed in User Layer requirements that they affect, they are not part of the kernel.

Darfin OODBMS supports fragmentation and location transparency. The User and Interpreter view do not necessarily need to know the exact location of processed objects. The objects and object classes may move between database nodes. It is possible to define a set of fragment copies that are used in a given transaction.

Darfin OODBMS supports replication. Replication may be either invisible or visible to the users. In the former case Darfin OODBMS is responsible of the total control of replication. In the latter case the User has control over replication.

5.2 Database aspects

The following database aspects are important when defining the requirements of IN-view and TMN-view.

- *Fault-tolerance.* A pure fault-tolerant database is always available. The internal behavior in a case of an unexpect event is hidden from the database users. The level of fault-tolerance in a system can vary from total fault tolerance that can deal with all hardware exceptions such as network failures and node crashes to software fault tolerance that can deal with problems caused by software failures such as programming errors.
- *Response time.* Response time means the total time that is needed to fulfil a single request in a database.
- *Throughput.* Throughput means the number of requests that can be processed in a given period of time.
- *Serializability.* Serializability means that concurrent transactions do not interfere with each other. In a serializable environment it is possible to construct a serial sequence of all requests that gives the same result than in the concurrent environment.
- *Exact reads.* An exact read guarantees that read information is up to date, both in time and in contents. For instance, in regular databases that use two-phase locking reads are always exact. In some other concurrency control methods it is possible that one process has updated data while another process still reads the old version of data Such a read is not exact. The

opposite of an exact read is a dirty read. In a dirty read it is possible that read information is not up to date or is partially updated.

- *Committed writes.* A committed write guarantees that the information is persistent and stable after the commit. An uncommitted write is invisible until it is committed. A non-committed write is persistent but is not guaranteed to be stable. For instance, in a main-memory database system it is not always possible to guarantee that all committed information is stable, although the probability that some information is not stable at a given time is very low.

For IN-view the database is a set of persistent objects the values of which may change rabidly in time. The most important database operations include number translations, accessing user profiles, and reading and writing mobility information. For IN-view the database aspects mentioned in the previous list have the following preferences:

1) *Response time.* Response time is the most important aspect for IN-view. For instance, the average response time for a number translation should be around half a second. An estimation of the translation procedure goes as follows:
   i) Service Switching Point: 100 ms
   ii) Network delay to Service Control Point: 10 ms
   iii) Service Control Point: 200 ms – 400 ms
   iv) Network delay to Service Data Point: 10 ms

   This leaves in average about 50 ms – 150 ms to the database access. In worst-case scenarios the processing time is much lower.
2) *Fault-tolerance.* An IN-view database should be fault-tolerant. The total down time in a year should not exceed ten seconds and it is allowed to be only fractions of seconds at a time. However, in extreme hardware failure conditions, such as network failures, it is acceptable that the database is unavailable for longer periods of time, such as minutes compared to seconds. Such problems are usually outside database control. Affecting them in the database architecture is not possible.
3) *Throughput.* Throughput is not as important as response time. IN-view transactions are often firm. The transaction result is not useful unless it is returned before the deadline of the transaction. This causes throughput to leave room for response times. It is acceptable that in busy conditions some transactions are not served or finished but all finished transactions are finished before the deadline.
4) *Committed writes.* A committed write is a preferable condition but not a necessity as long as the probability of loosing non-commit data is not high and the probability goes to zero in time. This allows faster writes and thus favors response time. The decision to use committed writes versus non-committed writes may be left to the requester. The system should offer both a secure committed write and a faster but less secure non-committed write.
5) *Serializability.* Serializability is sometimes important but in most IN requests it does not matter if the requests are serializable. It is much less important than response time, fault-tolerance, and throughput. Some requests in IN-view need strict serialization but in such cases the operations themselves are serial in nature (see Section 3.2).
6) *Exact reads.* It is acceptable that IN-view requests may read old data. A time constant defines how old read data may be compared to new updated data before the read is not accepted.

For TMN-view the database is a set of persistent objects attribute values of which remain relatively static in time. The database is mostly used for the customer information store, service maintenance, and network maintenance.

The accuracy of stored data is the most important aspect in TMN-view while the other aspects are somewhat lower in priority. This gives the following preference order:

1) *Serializability.* Serializability is very important in TMN-view. If the database cannot guarantee serializability it is possible that two simultaneous transaction interfere with each other and cause network information loss or inconsistence. Although this is not necessarily fatal it can be very costly.
2) *Committed writes.* Committed writes are very important. For instance, it is not acceptable that billing information is lost due to a non-committed write. TMN-view needs also distributed and scheduled commits. A distributed commit affects several nodes in a distributed database. A scheduled commit is activated in predefined time. For instance, updating routing information should be activated simultaneously in all updated nodes.
3) *Exact reads.* Exact reads are very important. For instance, when the software in a switch is revised it usually causes a lot of read and update transactions. If the transactions cannot guarantee exact reads it causes false updates to the switch. It is not acceptable that information is updated elsewhere during the transactions.
4) *Throughput.* It is preferable that all requests are answered in time. TMN-transactions are usually either soft or they do not have deadlines. The result of the transaction is still useful even if it is gained after the deadline of the transaction.
5) *Fault-tolerance.* Fault-tolerance is preferable but not as important as in IN-view. It may be acceptable that TMN-view is not accessible for short periods of time, such as minutes in a month compared to seconds in a year.
6) *Response time.* Most TMN-view requests are not dependent on time limits. Throughput is usually more important than the finishing time of a single request. although there may be exceptions to the rule.

In the previous list items 1–3 are about equal in preferences. Together they define the correctness of information.

IN-view and TMN-view have very different characteristics and preferences as can be seen from the preference lists. Although there are no direct conflicts between the items it is still difficult to build a database that can simultaneously fulfil both IN-view requirements and TMN-view requirements.

The IN-view is a real-time database where transactions are firm. The TMN-view is either a regular database or a real-time database where transactions are soft. Throughput and fairness are usually more important than serving a transaction in time. The TMN-view transactions must not interfere or interact with each other. Altogether TMN is a very complex environment and the requirements in TMN-view may vary in different uses.

5.3 Transactions and concurrency control

Transactions in Darfin OODBMS should have the following properties that are listed in [ElNa89].

1) *Atomicity.* Transactions are atomic. They are either executed completely or not executed at all. Darfin OODBMS supports atomicity.

2) *Consistency preservation.* If the database is consistent before the transaction execution then it will be consistent after the transaction execution as well. Darfin OODBMS supports consistency preservation.
3) *Durability or permanency.* Once a transaction is committed the changes made in it will not be lost because of subsequent failures. Darfin OODBMS supports durability and permanency to certain extent. In some situations, such as in IN-view 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.
4) *Isolation.* In strict isolation transactions should not make their updates to the database visible before commits, and thus transactions do not interfere with each other. There are several levels of isolation depending of the level of interference allowed. Darfin OODBMS supports isolation and different isolation levels. Objects may decide the isolation level they support. The lower the isolation level is the more the object is accessible. Also transactions may define the isolation level they want to use. In case of an isolation conflict (transaction wants lower isolation than is defined in the object) the higher isolation is chosen. Otherwise a parallel running high isolation transaction might not be able to gain the isolation level it expects.
5) *Serializability.* Transactions are serializable if the result of running them in parallel is equivalent of running them in some serial order. Darfin OODBMS supports serializability, though in some situations strict serializability may be unnecessary.

Darfin OODBMS must support the following transaction types.

1) *Transactions without deadlines.* Such transactions are usually used in maintenance Requests, either in IN-view or in TMN-view
2) *Soft transactions.* Such transactions are usually used in TMN-view in most Requests, but perhaps also in IN-view in certain Requests.
3) *Firm transactions.* Such transactions are usually used in IN-view in time-critical situations.
4) *Priorized transactions.* Such transactions are run according to their priority. A higher priority transaction has preference over a lower priority transaction.
5) *Uninterrupted transactions.* Such transactions are always finished.
6) *Non-wait transactions.* Such transactions are never put on hold (eg. a special case from priorized transactions where the transaction priority is the highest possible; what happens when two such transactions interfere is undefined.)

It is possible to define the transaction type in dynamically for each transaction.

A different characterization to Darfin OODBMS transactions is according to the subtransaction level. There are four different types of transactions (figure 3).

1) *Undivisible transactions (figure 3 a)).* Such transactions are undivisible objects that start at a given time and end when all the operations in the transaction are finished. There is no substructure in the transactions.
2) *Nested transactions with checkpoints (figure 3 b)).* Such transactions are divided into one or more subtransactions that themselves may be divided into subtransactions. All the subtransactions are executed in linear. If any of the subtransactions is canceled then the whole transaction is rollbacked. Every subtransaction may be considered a checkpoint to the transaction.

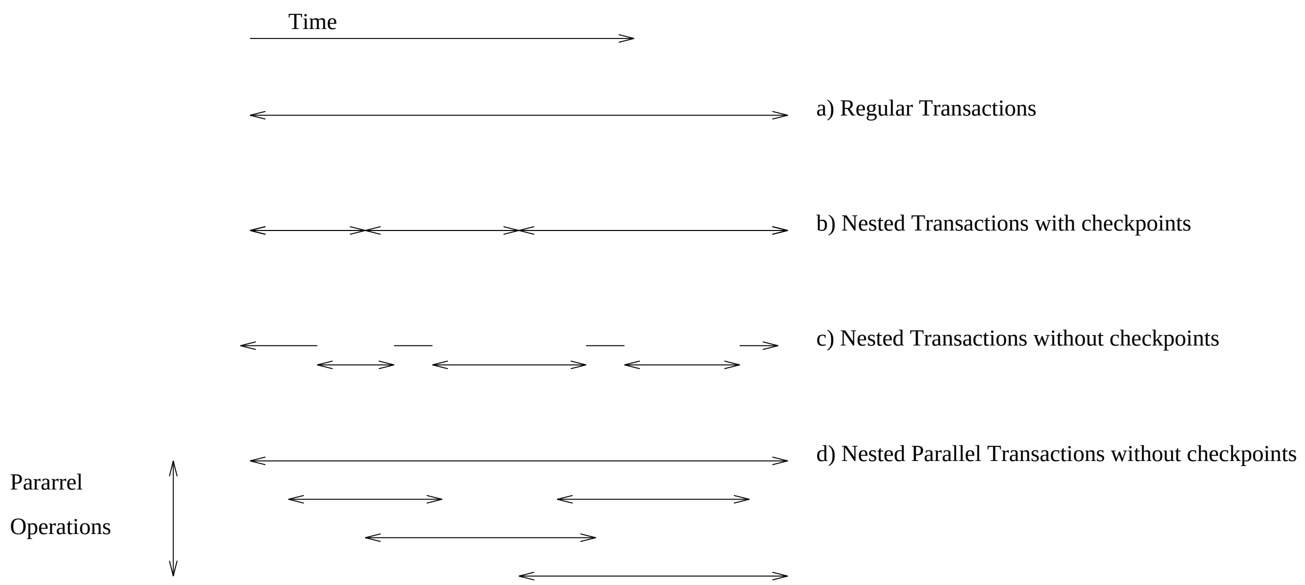

Figure 3: Different Darfin OODBMS transaction models

3) *Nested transactions without checkpoints (figure 3 c)).* Such transactions are divided into one or more subtransactions of arbitrary dimension, like in the previous class. All subtransactions are themselves independent entities that may be committed or canceled. The complete transaction may be committed even when one or more of the subtransactions are canceled.
4) *Nested parallel transactions without checkpoints (figure 3 d)).* Such transactions are divided into one or more subtransactions of arbitrary dimension, like in the two previous classes. However, the subtransactions may be executed in parallel.

All the previous classes are important in specific situations. It is also to be noted that the last class covers all the other classes as special cases.

Darfin OODBMS also supports nested distributed transactions. There may be parallel subtransactions in several Distributed Darfin OODBMS nodes.

A preferable condition for transactions is serializability. This is extremely important in maintenance functions, especially in TMN-view. Only conflict-based serializability is allowed, that is, the only way transactions interfere with each other is through resource conflicts. Darfin OODBMS is able to recover from aborted transactions and able to avoid cascading aborts.

5.4 Fault tolerance and recovery

Darfin OODBMS is fault tolerant, especially in IN-view where the total down time at a time may only be fractions of seconds. In TMN-view the fault tolerance requirements are not as firm. It must be possible to automatize fault tolerance and recovery control in Darfin OODBMS as long as both recognizing and isolating problems are possible. For instance, a node may not be able to recover if a total power shortage occurs in the are it is located.

5.5 Object operations

Darfin OODBMS is object-oriented. It implies that stored objects may have operations.

The object operations have the following properties.

1) Operations are part of the objects and they are stored in the database.
2) Operations are computationally complete. For instance, it is possible to build a language compiler inside an operation,
3) There are three different choices to execute an operation
   - In the caller's workspace
   - In the database's workspace
   - In an external workspace

   It is important to keep the security level in operation executions as high as possible.
4) It is possible to access other objects from operations, and other data structures that are known to exist at the operation execution time.

It is possible to have specialized object and database operations that may be triggered without a request from outside. The operation is performed if a predefined condition holds in the database environment. For instance, time is a condition that might trigger database operations.

Darfin OODBMS supports active object and database operations and offers a language for the maintenance staff to build such operations. The language supports at least the following triggering conditions:

1) *Triggers based on time.* Operations triggered by time are executed at a predefined time. For instance, a maintenance report may be generated periodically, or a global update might be performed simultaneously on several database nodes.
2) *Triggers based on the database state.* Operations trigged by the database state are executed when a predefined condition in the database holds. For instance, an alarm may be generated if a transaction performs an suspicious update to an object.

5.6 Other database and database kernel view requirements

Darfin OODBMS supports indexing to objects and object classes. At least the following index structures must be possible:

1) tree-based indexes
2) hash-based indexes

Darfin OODBMS supports at least the following online database operations in both IN-view and TMN-view.

1) Changing schemas and object classes
2) Reconfiguration
3) Reorganization
4) Scaling
5) Consistency checks and logging

Darfin OODBMS design supports extensions to the database operations, such as increment and decrement operations to numeric data. The exact set of database operation extension depends on Darfin OODBMS version. The Darfin OODBMS architecture supports adding new extended operations.

Darfin OODBMS supports database maintenance and monitoring from console.

6. Other requirements

This Section classifies such requirements that do not fit the previous structure of the Darfin OODBMS, or that affect more than one of the view layers.

6.1 Environment

It must be possible to run the Darfin OODBMS in the most common Unix environments, such as Posix, SVR4, HP-UX, and BSD 4.3.

7. Summary

In this report we have summarized the functional requirements needed for both IN-view and TMN-view. We have identified the different abstraction levels: User Layer, that defines the user view to the system, Interpreter Layer, that defines the possible interfaces to the database management system, Database Layer that defines the global database interface, and Architecture Layer that defines how the database interface may be implemented. Interpreter Layer, Database Layer and Architecture Layer together form Darfin TDMS. Database Layer and Architecture Layer together form Darfin OODBMS and Distributed Darfin OODBMS. Most requirements to Darfin TDMS are also requirements to Darfin OODBMS or Distributed Darfin OODBMS. All requirements raise from the User Layer.

In the User Layer we have defined the interface requirements needed for Darfin TDMS. Currently we have identified the following request sources: IN requests from SCF, IN requests from SDF, IN and TMN requests from external networks, IN CS-1 INAP requests from SCF, IN management requests from SMF, TMN management requests from MIS agents, IN and TMN ODP channel requests, high level query language requests from console, and interpreter bypassing requests from a trusted source. The Darfin TDMS architecture does not restrict the number of requests sources. It is possible to add new request types by adding new interpreters to the Interpreter Layer.

The Interpreter Layer offers a mapping from different User request types to the Darfin OODBMS. It is the high level application layer in Darfin TDMS. The interpreters may use either an interpreter request language that encapsulates database operations into a single transaction, or an embedded database language that offers a high level programming language interface to the database. The former gives more freedom to the implementation of Darfin OODBMS. The latter gives more freedom to the Darfin TDMS application programmers.

The Database Layer and Database Architecture Layer offer an interface and implementation to Darfin OODBMS. These levels together form Darfin OODBMS and its architecture. Also here the differences between IN use and TMN use are relevant.

Both in IN-view and TMN-view the data distribution level may be on the database kernel level, that is it is totally invisible to the users, or it may be in an application level and thus give control to the applications. The same is true to replication. It is usually completely invisible, but it does not have to.

The Darfin OODBMS architecture supports different object isolation levels, various transaction types including real-time transactions and parallel transactions and subtransactions, and active operations that may be triggered upon a predefined condition. It is possible to expand the indivisible database operation set that is used in transactions.

The data management architectures for IN and TMN differ from each other. The IN-view Darfin OODBMS must optimize response time and fault-tolerance in cost of throughput and serialization. The TMN-view Darfin OODBMS must optimize throughput and serialization in cost of response time and possibly fault-tolerance. This affects the general transaction types in IN-view and TMN-view. In IN-view the transactions are firm, while in TMN-view they are soft.

The conclusion of this report is that it will be difficult to build a real time data management system that may be fully used on both IN and TMN.

References

[ElNa89] Elmasri R. and Navathe S. B., Fundamentals of Database Systems. The Benjamin/Cummings Publishing Company, Inc., 1989

[Raat94b] Raatikainen K. E. E., Information Aspects of Services and Service Features in Intelligent Network Capability Set 1. Technical report, University of Helsinki, Department of Computer Science, 1994