Market Requirements Document

Document Control

Feature Name: Telecom Market Requirements Definition

Version:	3		February 11, 2000
Version	2		April 14,1994
Version	1		March 17, 1994
Version	0.1	1st. Draft	February 4, 1994

Submitted By: Leon Guzenda and Charles Johnson

Description of the Problem

Objectivity is determined to regain its position as the major ODBMS supplier to the telecommunications industry. This document summarizes the major concerns of the telecom companies we have worked with and the product enhancements or options required to maintain and strengthen our position in this market.

The telecom industry is still undergoing radical change. The emphasis is on providing service and features in an increasingly competitive business environment. Mergers and acquisitions and complete replacement or displacement of infrastructure are the order of the day. The growth is in services and new technologies, not in people or traditional hardware.

The hardware and networks are fault tolerant, while the software that runs the business is often not. System failures receive almost immediate national and international media coverage. As a result, the industry has committed to object-oriented technology to facilitate software maintenance and testing and to increase reliability. Many of the new technologies and services require support for arbitrary graph structures and multi-media. These types of data are more naturally modeled with an ODBMS.

Finally, the business, operations, management and maintenance systems for running the networks are being upgraded to provide interoperability and improved user interfaces. Most major carriers now provide some degree of web-enabled account and service management to their subscribers. Building and selling these systems generally involves conformance to existing and emerging standards and the definition of new standards. Luckily, most of the software standards are now object-oriented.

Please note that “Objectivity/DB” represents the current kernel and the C++, Smalltalk and Java language APIs throughout this document.

Description of the Requested Features

Overview

The major concerns of our telecom customers and prospects may be summarized as follows:

- Fault Tolerance
- Scalable Performance
- Interoperability
- Standards conformance
- New platforms/interfaces

Fault Tolerance

This is paramount for most network applications. The application and its data must be operational and available for the network to be functioning – failures are embarrassing and costly. The application is allowed zero down time. Critical data must be replicated (either physically or using RAID). Online synchronization of new replicas (e.g. to substitute for a faulty replica) must be possible. It should be possible to defer synchronization to partners separated by low bandwidth networks. It must also be possible to effect a very fast failover in the event that a "master" server fails.

The application must remain functionally active during a software release upgrade. For a short time after installation, it must also be possible to roll back an upgrade. It is acceptable to choose the temporary rollback capability at the expense of redundancy. The standard telco practice is to split a redundant system and upgrade one side. If performance is acceptable after a day or so the system is reunified with the new load, otherwise the system is reunified with the old load. We must provide mechanisms and guidelines for online migration of applications, without loss of service or data. Finally, a mechanism equivalent to a safe message store or a rollforward log is required to prevent the loss of critical messages or transactions. This may be an external mechanism with supported interfaces to/from Objectivity.

The exact mix of requirements depends on the individual application. For instance, it may be tolerable to lose a few Call Detail Records per million, but it is not tolerable to lose a stored fax or voicemail; or to lose the routing for a 911 call or a major 800 number subscriber. This translates to a range of options geared to the availability of hardware and operating system resilience features.

Objectivity must introduce and sell telecom oriented support options, such as instant response and long term support for deployed versions. Willingness to commit to telco style maintenance and then actual delivery of the services required will lock us into a dominant position as the telco embedded database of choice.

In summary, the new features to support improved fault tolerance are:

- Low bandwidth synchronization of replicated data
- Support for online software upgrades
- Safe transaction queueing
- Easily configurable fault tolerance options.
- Very fast failover.
- Instant response and deployed version support options

Scalable Performance

Network Data Model – The distributed design of Objectivity/DB is extremely well suited to many telecom requirements. Data created locally by a logging or scanning process can be kept locally except when required for detailed analysis. This allows capacity to grow naturally with a network as nodes are added, and minimizes the impact of network channel capacity and latency on applications. For this to be practical each node must be a partition with its own lock server and it must be possible to add and delete nodes from a network even if some nodes are down or inaccessible. It may be worth providing a generic tree/network iterator (transitive closure).

Short Transactions - Objectivity/DB was designed to handle data intensive transactions. Each transaction is expected to last seconds rather than milliseconds and is expected to manipulate large quantities of complex data residing in massive databases. RDBMSs are generally designed to handle short transactions that return relatively little data from a medium sized data set. Very few telecom applications, outside of simulation and analysis applications, use large quantities of data in a single transaction. They are generally message based, often having to respond within a small slice of the call setup time. Luckily, Objectivity/DB can cache objects that are primarily read-only, providing in-memory speeds for many critical transactions. However, its overheads for updating a single object in a transaction are extremely high compared with conventional server-centric architectures. Objectivity/DB should be configurable to the extent that it can support short transactions with minimal overheads. Realistically, updating a single small object in one transaction should involve no more than three I/Os to find the object (zero if it is already cached) and three or less I/Os to perform an ACID update of the contents of that object.

Scalable Throughput - Telecom applications may have to support hundreds of concurrent connections to other pieces of equipment and millions of active “transactions”. It is vital to exploit multithreading wherever possible. The client application should fully support standard multithreading models. The servers (AMS and Lock Servers to date) should also be multithreaded wherever this reduces response times or increases server throughput.

Physical Scalability - Telecom applications do not generally produce databases as large as the ones generated by simulation or data acquisition applications. However, it is important to be able to split a federated database across many drives or sites, mainly to improve fault tolerance or local performance. Databases should be able to grow to the maximum size permitted by the underlying operating system and the federation should be able to support four billion (2 ^ 32) files rather than the sixty four thousand (2^16) it supports now. As a simple example of the latter, if any local carrier built a system that dedicated a container per subscriber and that container had to live on the subscriber’s PDA or wireless phone then current federations would soon run out of files.

Low Bandwidth Networks - Objectivity/DB uses a distributed processing architecture. It uses more CPU cycles in the client than a traditional DBMS and it relies on reasonable bandwidth and low latency between the client and server. Reasonable means that remote I/O should generally cost no more than about 30% more than local I/O. The minimum unit of transfer is a page, which defaults to 8 Kb. This would clearly not provide interactive response over a 28,800 Baud modem line and we should probably not focus our efforts on supporting that model. However, separating the client API from its directly linked interface to the rest of the kernel would allow a user to perform many operations, particularly queries, across a low bandwidth line. In effect, Objectivity/DB could be configured to compete head on with Versant, POET and other server centric architectures in low bandwidth situations.

High Latency Networks – Satellite and other wide area networks often suffer from high latency. It is expensive to setup a message interaction, but the message may be supported by very high bandwidth. Recent measurements by a Japanese team collaborating with CERN showed the impact of this latency compared with operation within a LAN. The measurements clearly show that the latency involved in setting up each interaction between the client and the AMS completely dominates the perceived I/O performance. Objectivity/DB should be configurable to use a protocol (probably asynchronous) that can move data in larger chunks to mitigate the cost of the setup time. This should work for both reads and writes, e.g. with the AMS transmitting all of the pages of a container as rapidly as possible to a kernel that buffers the received pages until the Storage Manager needs them.

Monitoring – Objectivity/DB has comparatively few performance monitoring and tuning aids. There are some client side statistics (ooRunstatus) and a Lock Server monitor. It is difficult to monitor the effectiveness of an application's caching, indexing, hashing, clustering (including containerization), locking or transaction management strategies. Objectivity/DB should include:

- Extra statistics related to cache management
- A clustering analysis tool (summarizing the clustering patterns within a container/ database/FD and their frequencies)
- Better server monitoring tools (possibly including hooks for SNMP)
- A better tool for analyzing physical space utilization (a sanitized oospace).

Interoperability

New telecom applications rarely exist in isolation. They either have to interface with existing equipment or with legacy applications. This section explores several of the facets of interoperability.

Gateways – It is frequently necessary to pass data to and from legacy applications. This can sometimes be done by extracting files of data or by using RDBMS triggers or snapshots to send data to the OO application. Conversely, relational tools can use the Objectivity/SQL++ ODBC interface to extract, create or update objects (albeit at very low bandwidth compared with the native C++ or Java interfaces). Objectivity should provide standard guidelines for implementing common RDBMS to Objectivity data gateways.

OO applications can generally call either Objectivity/DB or an RDBMS by proper use of encapsulation. Some products also provide gateways to RDBMSs via the ODBMS API. Objectivity/DB is a Federated Database, but it currently uses a single Storage Manager to manipulate objects. We have never provided an OSI Remote Database Access gateway to channel objects between the Object Manager and a Third Party DBMS. This could be done but it can probably still be regarded as having a very low priority relative to all other requirements in this document. The logistics of integrating and QA'ing such an interface make this an expensive feature to engineer and maintain over long periods as the Third Party Products are still evolving.

Distributed Transaction Management – if it is necessary to co-ordinate update transactions across Objectivity/DB and another DBMS then it is generally easiest to employ a Distributed Transaction Manager (DTM). The ISO/OSI XA standard describes the protocols involved and there are several commercial implementations of XA, including Encina. Other products, such as Tuxedo and Weblogic, also exploit XA protocols. Objectivity/DB manages atomic transactions across distributed and possibly replicated databases. Its Lock Servers could become XA Resource Managers co-ordinated by a DTM. This should be an optional feature of Objectivity/DB.

Security – If any system is accessible from another system then security becomes an issue. Objectivity/SQL++ implements the standard GRANT/REVOKE model for SQL++, but this does not affect access via other APIs. The Generalized Security Architecture used within Objectivity/oofs applies to files (databases) and not to containers or objects. Luckily, most telecom applications are shielded within a proprietary environment or use standard, reasonably secure, messaging interfaces. However, there is increasing concern over the inability of the ODBMS to enforce security constraints. This is a large enough topic that it merits a separate MRD. For now, we should note that Objectivity/DB should implement a standard security model that could be applied to at least containers and preferably objects or collections of objects.

CORBA – We have published White Papers and can provide examples of ways of integrating Objectivity/DB with CORBA products, such as Orbix (C++) and Visibroker (Java). We should ensure that these examples are kept current with the emerging CORBA standards and products. We should also re-open our marketing links with the major vendors. There should be optional products for cleanly migrating IDL to DDL and vice versa. It may be sufficient to ensure that a viable roundtrip engineering path is available using Rational ROSE and Objectivity/ROSElink.

Message-based APIs – It comes as a surprise to most telecom developers that Objectivity/DB is not a message based (client-server) product. It would be very useful to have:

- An optional lightweight client that uses a message–based API (separating the Language Interface and Object Manager, mainly, within the product) and the corresponding server (the rest of the kernel).
- Event Notification (productizing and enhancing the current library built by Todd Stavish, e.g. with an asynchronous trigger mechanism)
- A shared, secure ("committed") object queue class library. This would be extremely useful in a wide class of telecom applications and it could be saleable in its own right.

Standards Conformance

Languages - Historically, Objectivity has played an important role, relative to its size, in the standards community. We have participated in ECAD (CFI), MCAD (CALS, STEP and PDES), GUI (X and OSF) and OS (X/Open and OSF) standards groups or consortia in the past. We currently participate in DBMS (ODMG and ANSI SQL) and telecom (NM Forum) standards groups.

Objectivity/DB uses ANSI standard languages and supports the vast majority of features of those languages. The notable exceptions are unions, object arrays and pointer arithmetic in ANSI C++; Internationalized Java strings; and collections, ODL and OQL in the ODMG language bindings. We should clearly document these omissions and our reasons for non-compliance. In general, most of them can be ignored. We should solicit feedback from our users on these omissions and fix any that cause major grief to programmers.

Internationalization – Our product uses POSIX message catalogs and our distributor in Japan supports a localized (Kanji) version of Objectivity. However, Objectivity/DB does not support Unicode or 16 bit indices, hash tables, predicates or SQL++. Neither does it support local collation sequences. All of these items should be addressed. It may be reasonable to support only single language operation within a given federation, database or container, otherwise, in an extreme case, every string would need a language stamp. Collating output containing mixed languages is, at best, interesting, so we could support a single language by default and provide hooks to allow users to mix languages.

Industry Standards – We loosely support CORBA, but it would be good to be able to generate DDL from IDL and vice versa. We should optionally support the XA distributed transaction protocol. We should issue a formal statement on our compliance with the latest version of the ODMG interfaces and on our position relative to the various NM Forum software platform models. XML is addressed in the MRD on “Internet Friendly Features”.

New Platforms

Objectivity/DB supports most of the platforms currently used by the telecom industry, with the notable exception of MVS, VM and OS/2. We have supported some fault tolerant machines, such as the IMP and the Stratus, for as long as we had a customer base for them. It is unlikely that anyone will specify that we must run on any of the proprietary IBM platforms, though it could be done on a paid for basis. It is much more likely that a trimmed down real-time version of Objectivity, currently known as Objectivity/RT, would be useful to the telecom industry, particularly builders of switches, network elements and handheld devices. There is a separate MRD for this product. Possible platforms include Java/RT, Chorus, pSos and vRtx.

Part of an existing feature or does it require another feature, if so, which one?

Please refer to the previous section for details. Most of the features would be offered as standard features of Objectivity/DB. Objectivity/RT has its own MRD. The following features could be offered as options:

- Fault Tolerance/Replication features would be added to FTO/DRO.
- Low Bandwidth Option (A.K.A. Message Based API)
- High Latency Option (Objectivity/Satellite)
- Gateways
- Distributed Transaction Management
- Any security feature that incurs Third Party licensing costs on our part
- CORBA interface

There should be a new or updated MRD for each of the above features.

How is this problem being solved now, and why isn't that acceptable?

In general, our users are:

- Working around the lack of these features
- Building their own persistence solutions
- Choosing other products.

What languages must support this capability?

The following APIs should support the new features:

- C++
- Java
- Smalltalk, but only where the feature can work within the existing API.
- SQL++

Which platforms must be supported?

All current platforms. SGI Irix is probably the least used in online telecom applications.

Do any competitors already have this feature?

- POET has better ODMG (3.0) compliance.
- ObjectStore has better ANSI C++ conformance.
- POET and Versant perform better in low bandwidth networks.
- Versant has active links with Third Party telecom class library vendors, notably Vertel and DSET.

Customers who require this feature

- VARs and ISPs working in the telecom industry, such as Qualcommand Nortel
- Low bandwidth or high latency support is primarily required for eCommerce

Revenue at risk, or which could be won

During the past year Versant alone has announced multi-million deals with Ericsson and France Telecom. We would have stood a much better chance of being involved if we had been clearly recognized as the ODBMS of choice for the telecom industry. They also beat us at Siemens and in an unknown number of deals associated with their partnerships with Vertel and DSET. We are often out-featured.

When is this required?

- Mid/late 2001

Additional Notes

We will also need:

- Marketing collateral and a telecom seminar
- Qualification questions/notes for each market sector
- Sales training material
- A list of publications and industry events to be approached/attended
  Investment in the staffing and training required to implement the telecom oriented support options.
-