Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

November 1994 Telecoms ORB Workshop Report

Andrew Herbert

Abstract

This is a summary report of a one day workshop held alongside the November 1994 ANSA Technical Committee meeting, to discuss the requirements upon and likely structure and functions of a “Telecoms Object Request Broker”.

Specifically the workshop identifed weaknesses in the current OMG standards for ORBs and Object Services against those requirements, and in terms of extensibility to add desired additional features.

The concept of a “Framework ORB” as a compact, efficient skeleton into and onto which different protocols and functions could be loaded to suit different environment was identified and explored.

A number of project involving ANSA sponsors relating to the overall topic were introduced.

APM.1388.01 **Approved** 12th January 1995

Project Management (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

1 Introduction

This is a summary report of a one day workshop held alongside the November 1994 ANSA Technical Committee meeting, to discuss the requirements and structure and functions of a “Telecoms Object Request Broker”.

Specifically the workshop identified weaknesses in the current OMG standards for ORBs and Object Services against those requirements, and what extensions are needed to these standards.

The concept of a “Framework ORB” was explored. A Framework ORB is a compact, efficient skeleton into, and onto, which different protocols and functions can be loaded to suit different environments.

A number of project involving ANSA sponsors relating to the overall topic were introduced.

The report is laid out as a series of sections covering the main themes discussed. It is not a transcription.

The following presentations were given and copies of slides can be obtained from the authors:

- Andrew Herbert (APM) - Critique of CORBA and related standards
- Paul Vickers (HP) - Requirements
- Patrick Farley (BT) - COMBAT Broadband Services - enterprise model and technical infrastructure
- Francois Horn (CNET) - Retina - A project to design and build a TINAC DPE
- Cosmos Nicolau (Nemesys Research) - DCAN - a project on ATM signalling for dumb devices

2 Requirements

- The need for a Telecoms ORB arises from the trend to apply open systems in telecoms, especially by the newer operators and networks.
- The Superhighway and associated concepts (Information Supermarket, Telepresence, etc, etc) are drivers for opening up networks to third parties; providing access requires interoperability. Portability, speeds up deployment and hence is a competitive advantage.
- The requirements for a Telecoms ORB in network management and switching are similar to those for a “Real-Time ORB” in Test and Measurement, in Medical systems and in Financial systems.

Table 2.1: Telecoms ORB characteristics

Application Area	Extensions needed over current ORBs	Added value	Underlying Operating System
Interactive multi-media services	Streams, explicit binding, QoS negotiation, trading ease of adding / connecting to third party software scaleability, continuous availability, security	Greater interoperability between services stable APIs for portable implementations modular replacement / extension decrease software costs and reduces time to deployment	High performance Unix-like
Network Management	Integration of GDMO, CMIS, SNMP etc scaleability, continuous availability, security	Automated management of the network reduces (people) operating costs	Unix-like
Switching and transmission control	Optimized subset (e.g.) no repository high throughput (5000 TCAP instructions/sec) scaleability, continuous availability	Interworking with network management and service systems unified communications interface over diferent signalling protocols	Real-time microkernel

3 Weaknesses in current ORBs

- Transparency cannot be (selectively) turned off - demanding applications cannot control resources to guarantee throughput and predictability, etc.
- Policies are predetermined and built in - applications cannot negotiate with their infrastructure to optimize their behaviour.
- Object services define a stable API, but do not simplify the programming burden: 1173 “classname.method” calls are as hard to remember as 1173 function calls. More should be done to
  - — exploit generic functions (without resorting to the “Any” and “String” types, so that maximum static checking is maintained)
  - — hide book-keeping behind e.g. declarative IDL extensions, ODL wrappers for applications code).
- Interface Repository is under specified (and not always necessary)
  - — use of type names in “Any” and DII/DSI either implies a global repository, or that federation of repositories is solved within the ORB.
  - — global repositories require prior agreement on name space structure and ownership; this may not be possible in a deregulated environment
  - — the implementation of repository federation itself requires ORB functions - where does the recursion stop?
  - — repositories are only needed in systems which are extended with new types and where “type browsing” is required.
  - — a programmatic interface to the repository is needed - programs want to determine type codes, not IDL.
- Type conformance is undefined:
  - — many implementations assume name equality is best - this brings an overhead of registering types and exposure to management bungles
  - — repository defined subtyping is even more fragile
  - — server defined subtyping (as in Microsoft’s COM) defers checking until runtime, and leads to long term clutter in servers as they try to support previous versions of interfaces
  - — ODP style signature matching overcomes these problems, and sensible use of trading properties minimizes risk of “false matches”.
- Concurrency is poorly supported (when to use a factory versus when to fork a thread).
- Garbage collection (i.e. reference management) is missing leading to resource leaks.
- Visualization and debugging tools are missing.
- There no no benchmarks except basic RPC times. What other metrics apply?

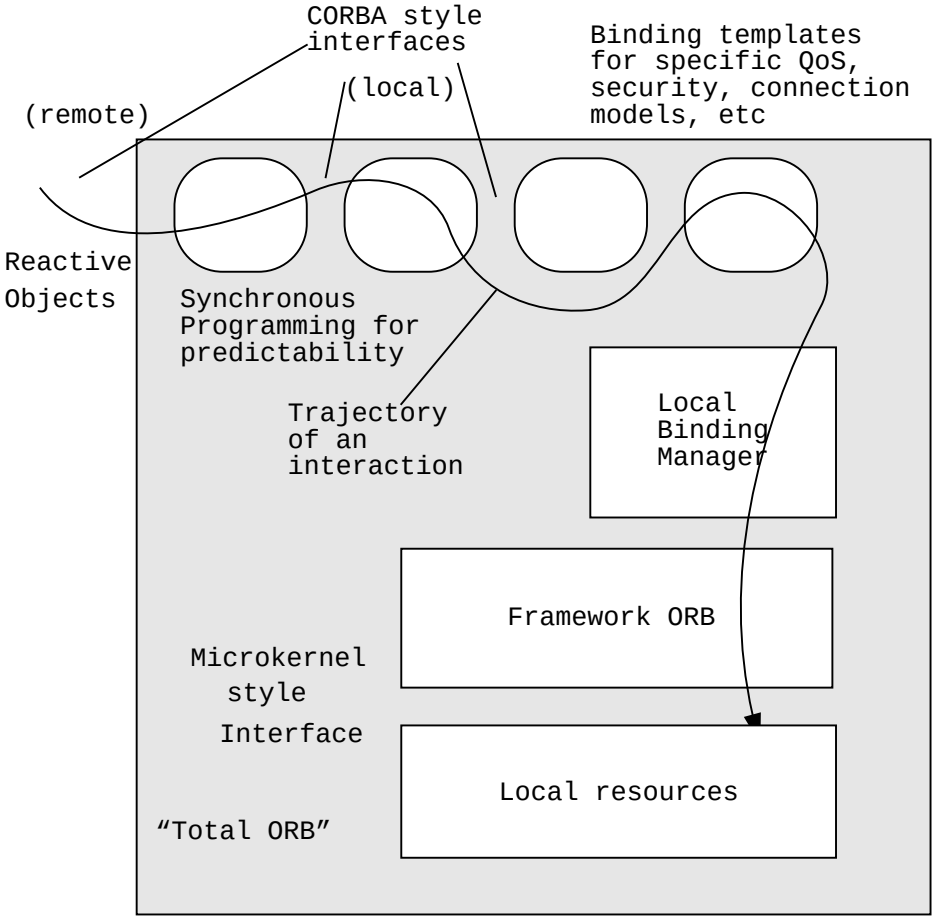
4 Telecoms specific issues

- Where does CMIS & GDMO fit in?
  - — can we hide them transparently behind ORB invocation and IDL?
  - — should we plan for native CORBA interfaces on managed objects replacing CMIP interfaces?
  - — what monitoring/ management interfaces do we need into ORBs?
- How does binding relate to connection management?
  - — binding implements connection management, if not provided by underlying (switching) technology.
- Do “signals” and synchronous programming allow thread and object models of computation to match efficiency of traditional telecoms event driven style?
- How should ORBs adapt to smarter networks? What is the best trade-off between end system versus switch/router functionality in
  - — multi-service communication (e.g. ATM integration of video, audio, data transport)
  - — service integration (e.g. HTTP encapsulation of FTP and Telnet)
  - — group management, multi-casting
  - — resource reservation
  - — content-based routing for control, policing, security.
- Simple devices at the edge of a network cannot handle complex signalling schemes (e.g Q2931, UNI series, SPANS, etc). They will be looked after by proxy managers which will use RPC to talk to one another (devices will be controlled at the cell level). Managers need
  - — secure delegation
  - — proxy transparency
  - — out of band communication (control goes via proxy but data directly to/from the device).

5 Framework ORB

- Supporting many speciaized ORBs is difficult for a vendor, ideally want an ORB that is
  - customizable to different markets
  - can be stripped down for ue in embedded systems
  - portable across Unix, real-time and microkernel operating systems
  - supports different RPC and thread implementations (for integration with other environments, such as DCE, Taligent, ...)
  - interfaces to other protocols (TCP, OSI)
  - provided with local object optimizations.
- Core ORB functions:
  - managing object (interface) references
  - managing local bindings
  - resource management hooks from application object model to underlying OS.
- ORB “plug-ins”
  - resource managers for specific management policies
  - binders / connection managers (including locators, traders etc).
- View the network as a “binding provider”, where a binding is defined in terms of end-to-end pattern of connectivity and quality of service parameters
  - temporality
  - volume
  - security, reliability, cost/billing
  - join / leave rules.
- If we make core ORB really efficient can we implement streams and invocations at the same level (i.e. define a minimum communications stack)?
- A “interface” is represented by:
  - a local object (stub) for access, maintaining both the binding and QoS (the stub links local scheduling and buffering to network events)
  - a local object (monitor) for observing QoS, resources etc
  - an interface reference.

The following figure illustrates the concept. A “Framework ORB” is show support in local binding manager and set of specific binding objects to sustain remote interaction. From an applications perspective, these components are all within the “ORB”. Their function is to add particular QoS management and



STRUCTURE OF A TELECOMS ORB

transparency features to interactions. These components are likely to use techniques such as synchronous programming to permit end-to-end calculation of qualty of service.

6 Kinds of Telecoms ORB

The following classification is not exhaustive and there are clear overlaps. The purpose is to give a feel for the range of configurations required.

- minimal ORB (foundation of the others)
  - local binding
  - interface reference management
  - basic management of threads, virtual memory, IPC/lightweight RPC, clocks
- switch ORB - the network
  - (simple) resource managers (assume a very static configuration)
  - remote binding
  - real-time, deterministic
  - dependability inherent in the application
- network provider ORB - managing the network
  - complex resource managers, including connection management (i.e. specific binding templates)
  - basic trading and life cycle management (although configuration wil be mostly static - i.e. changed on human timescales)
  - internal security and dependability (transparent to the ORB)
  - monitoring / accounting - implies link to databases
- service provider ORB - using the network[1]
  - connection management
  - federated trading and life cycle management
  - end-to-end security, dependability
  - tarrifing (“a service is anything that can be tarrifed”)
- end-user ORB (ORB in a handset or desktop box) - using services
  - object mobility (c.f. telescript) to function shipping, supporting agents, downloading etc
  - multi-media capable, out of band stream management
  - end-to-end security, dependability
  - service broking
  - intermittent connection, rapidly changing configuration
  - interface to desktop systems

1. N.B. this ORB is interfacing the service providor to the network - it may be that the providor uses “data processing” ORBs (c.f. “Telecoms” ORBs) internally.

— interface to embedded systems (e.g. in the home).