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

Slides: Distributed Interactive Multimedia Architecture

Guangxing Li

Abstract

This talk (Dec. 94 TC) briefs the design progress, the infrastructure built so far, the work strategy and the goals of the ANSA Distributed Interactive Multimedia Architecture.

Because of the similarity of the ANSA and CORBA object model, this talk can also be regarded as “how to extend CORBA for real-time and multimedia processing”.

APM.1360.00.01 **Draft** 5th December 1994

Request for Comments (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

ANSA Distributed Interactive Multimedia Architecture

(or how to extend CORBA for real-time and multimedia processing)

Guangxing Li

(gxl@ansa.co.uk)

In this session

- design progress
- infrastructure implementation
- work strategy
- the goals

Not in this session

- business case, motivation and benefits
- technological challenges
- scope and issues to be addressed

---> read APM.1295 (discussed last TC)

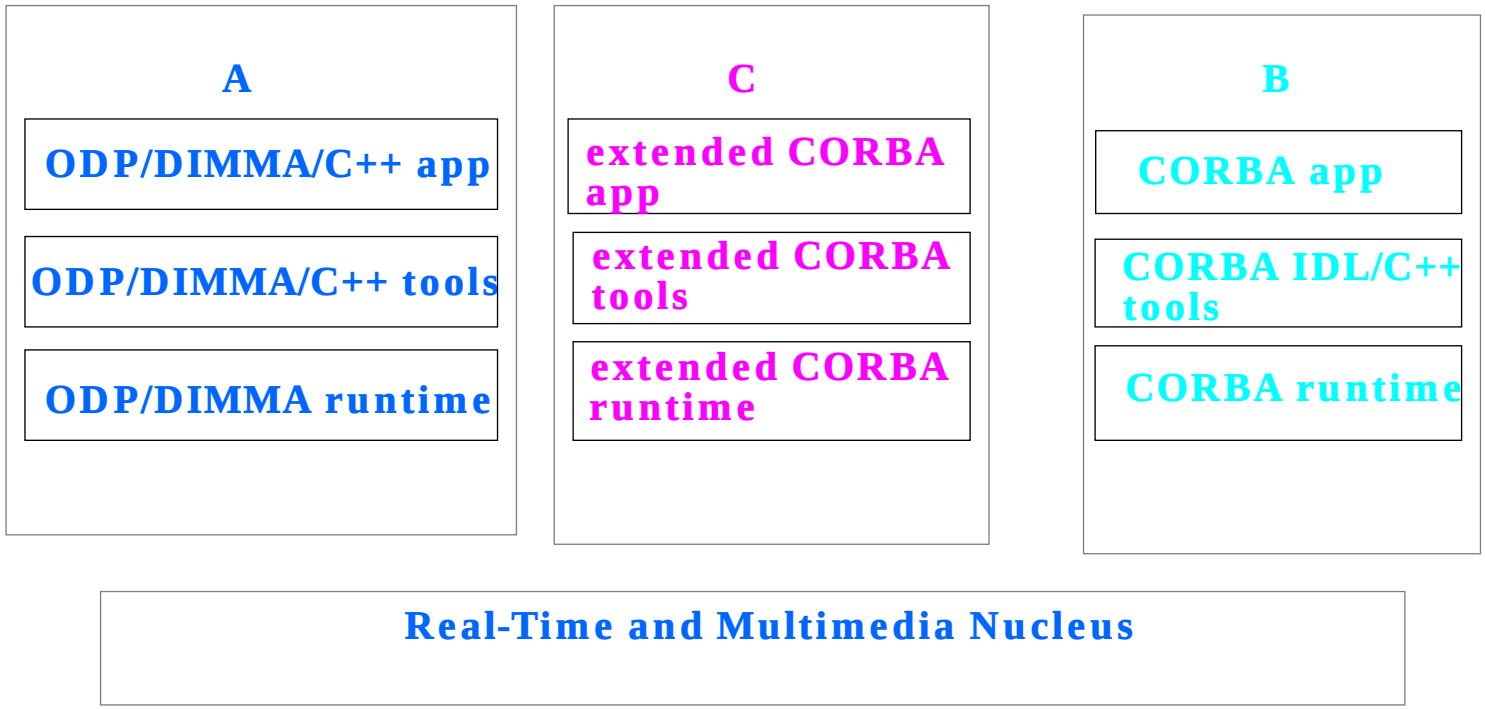
Design for multimedia and real-time processing

- interaction model: client/server (many-to-one) + end-to-end (stream)
- invocation model: (RPC) call/reply style + (message-oriented) signal passing
- control model: asynchronous + synchronous
- binding model: implicit + explicit
- QoS model: addressing non-functional requirements
- real-time programming model: introducing priority & deadline to distributed processing (or what is a real-time object)
- engineering model

Infrastructure implementation

- **ANSAware/RT over OSF/1, HP/RT, LynxOS**
  - **resource separation and independent scheduling**
  - **real-time and time-sharing scheduling**
  - **binding engineering**
  - **Quality of Service engineering**
  - **timed RPC**
  - **flexible multiplexing**

The next steps



Architecture goals

- follow ANSA principles
- maximum compatibility with RM-ODP
- maximum alignment with CORBA
- clear separation of programming model from engineering model

Programming model goals

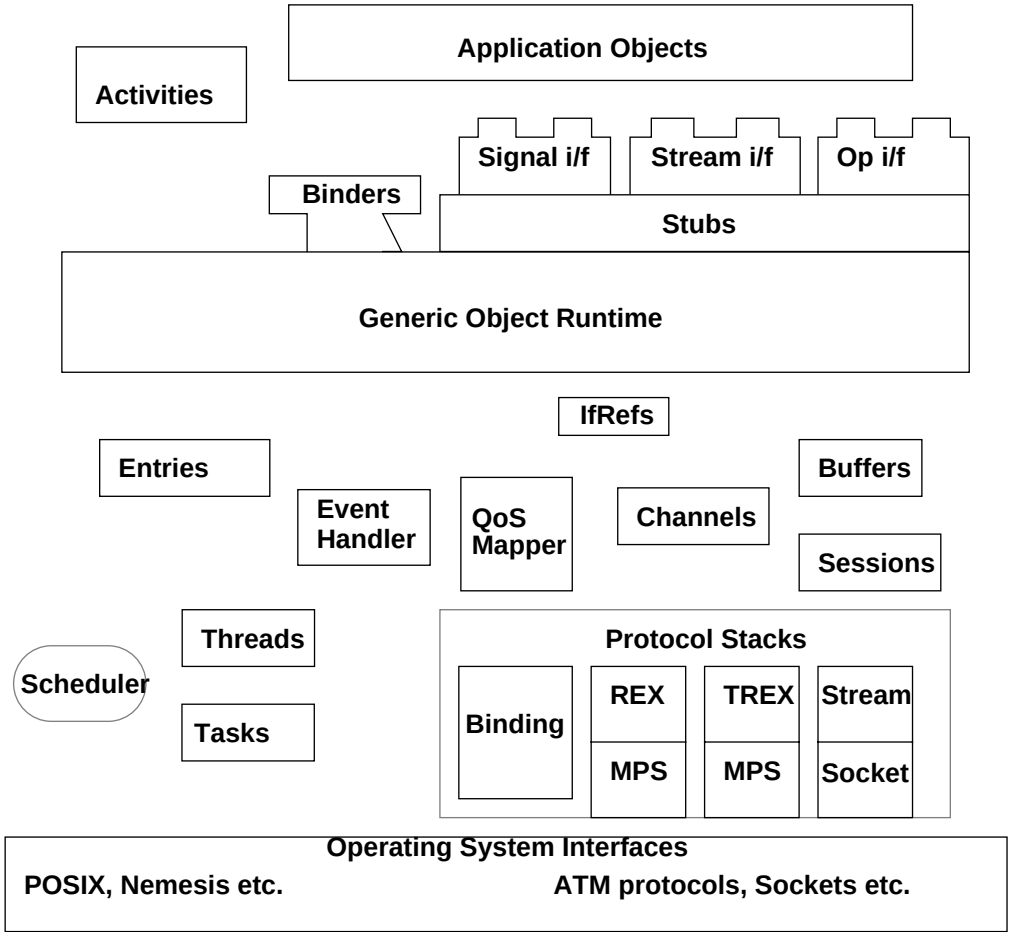
- a complete set of constructs for the extended computational model
  - explicit and implicit binding
  - synchronous and asynchronous expressions
  - operational interfaces and stream/signal interfaces
- strong type checking
- access and location transparency
  - enable local optimization
- selective resource transparency
  - controlled scheduling
  - controlled communication multiplexing
- imperative specification of QoS directives

Engineering model goals (1)

- generic framework (interface) enabling different implementations and implementation tradeoffs are possible
  - modular - easy to replace alternative components
  - extensible - easy to add extended functionality
  - scale up (for large applications of many clients and servers) and scale down (for embedded applications which have limited resources)
- scalable and resource-efficient implicit binding
- QoS driven explicit binding
- generic communication architecture for multiple protocol stacks, addressing schemes and communication models
  - x-kernel (UBC and UA) and ADAPTIVE Communication Environment (WU)
- different execution protocols for different style of object interactions

Engineering model goals (2)

- generic QoS specification, conformance check, negotiation and monitoring
- resource separation and independent scheduling
- both real-time and time-sharing scheduling
- allow to map onto any suitable real-time and multimedia technology
- end-to-end communications management
- object-oriented concurrency (fit C++)
  - u-C++ (U.Waterloo), SyncC++ (EPFL, switzerland), ESTEREL
- open to alternative APIs (e.g. specialized languages, preprocessor, library, CORBA etc.)


Application Objects
Activities
Signal i/f
Stream i/f
Op i/f
Binders
Stubs
Generic Object Runtime
IfRefs
Entries
Buffers
Event Handler
QoS Mapper
Channels
Sessions
Threads
Protocol Stacks
Scheduler
Binding
REX
TREX
Stream
Tasks
MPS
MPS
Socket
Operating System Interfaces
POSIX, Nemesis etc.
ATM protocols, Sockets etc.

Implementation goals

- maintainable - a common coding/documentation standard
- provide easy-to-use development tools
- using object-oriented design and particularly C++ language for programming
- maximum reuse of CORBA implementation(s)
- maximum reuse of ANSAware ideas
- good raw performance
- use industry standards wherever possible (CORBA, POSIX, ANSI, ITU etc.)

Current state

- **C++ coding templates for the programming model**
  - objects, interfaces, operations etc.
- **a multiple platform development tool set**
  - an integration of the GNU development tool set and ANSAware development tool set
  - added functionality and reduced complexity
- **in progress**
  - nucleus design and implementation
  - other distribution tools

CORBA issues --- tomorrow's workshop !