Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

Slides for Real-Time QoS

Guangxing Li

Abstract

Slides of RC.1151 “A Model of Real-Time QoS”.

APM.1157.00.01 **Draft** 28 February 1994

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

Distribution:
Supersedes:
Superseded by:

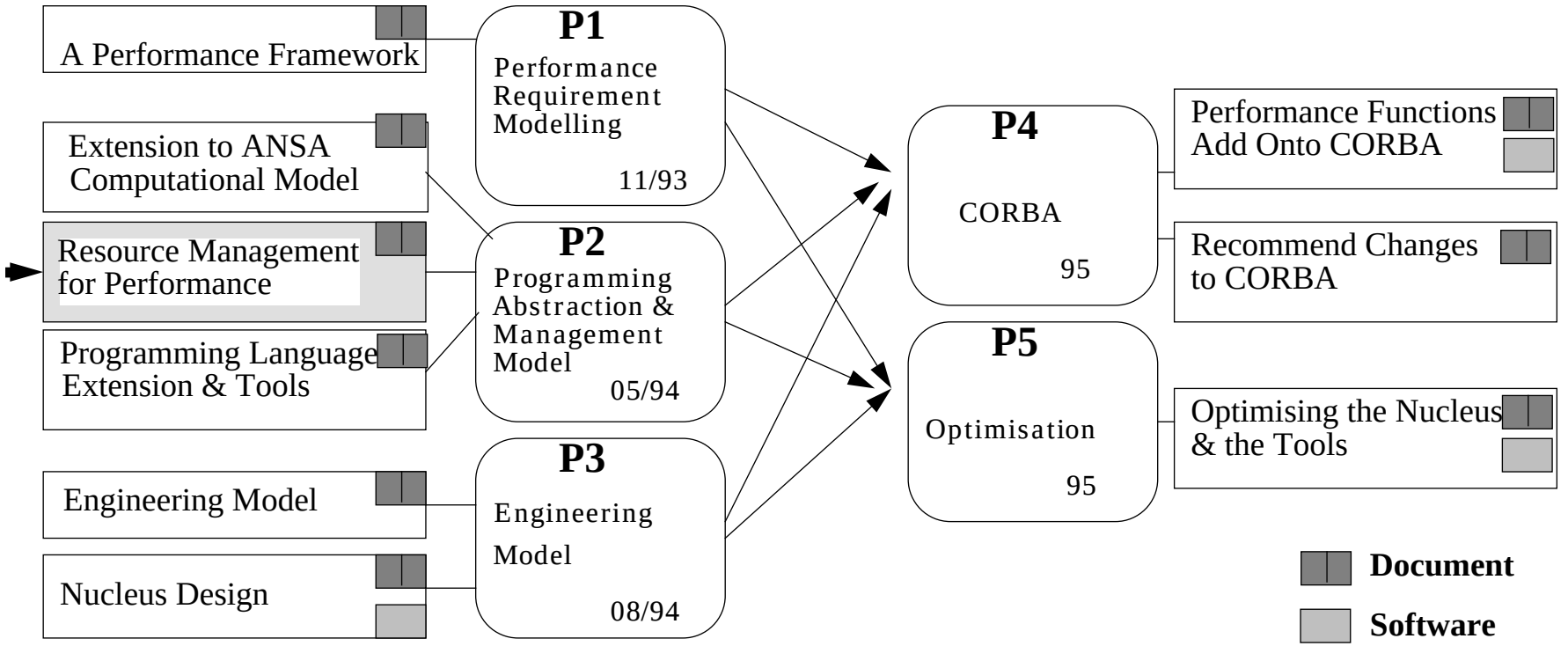
A Model of Real-Time QoS

Guangxing Li

gxl@ansa.co.uk

Performance Task Group

Group Activities



Objectives, Benefits and Context

- **Objectives**
  - **identify real-time QoS specification issues: requirements, functions, design constraints**
  - **provide a QoS framework: concepts, entities, tools and their structures**
- **Benefits**
  - **explains how the open system (ANSA) concepts are applied to real-time systems**
  - **explains how real-time technologies are integrated with open systems**
- **Context**
  - **RC 1072: Engineering Aspects of Real-Time**
  - **RC 1150: Towards an ANSA Binding Model**
  - **RC 1151: A Model of Real-Time QoS**

QoS Characteristics

- **QoS is a generic mechanism to express**
  - **the provided performance**
  - **the required performance**
  - **the required resources for the provision of a service**
  - **the required resources for the access of a service**
- **QoS characteristics**
  - **categorised: as QoS domains, e.g. communication QoS domain, multimedia QoS domain.**
  - **modelling behaviour and/or resource constraints: declarative or imperative statements.**
  - **quantifiable: itemized to individual quantified parameters (statically-bound or deterministic).**
  - **combinative: QoS items can be combined.**
  - **advisory: hints in the selection of underline mechanisms and policies.**

QoS Specification

- **Requirements**
  - **hiding engineering details which are irrelevant to application programmers**
  - **hiding arbitrary syntactic differences between technologies**
  - **keeping concepts as independent and orthogonal as possible**
  - **imposing the simplest possible conformance rules**
  - **providing the maximum negotiation flexibility**
  - **providing the maximum configuration flexibility**
  - **practical and prescriptive**
- **Approaches**
  - **Application Programming Interface (API) based approach**
  - **language-based approach**

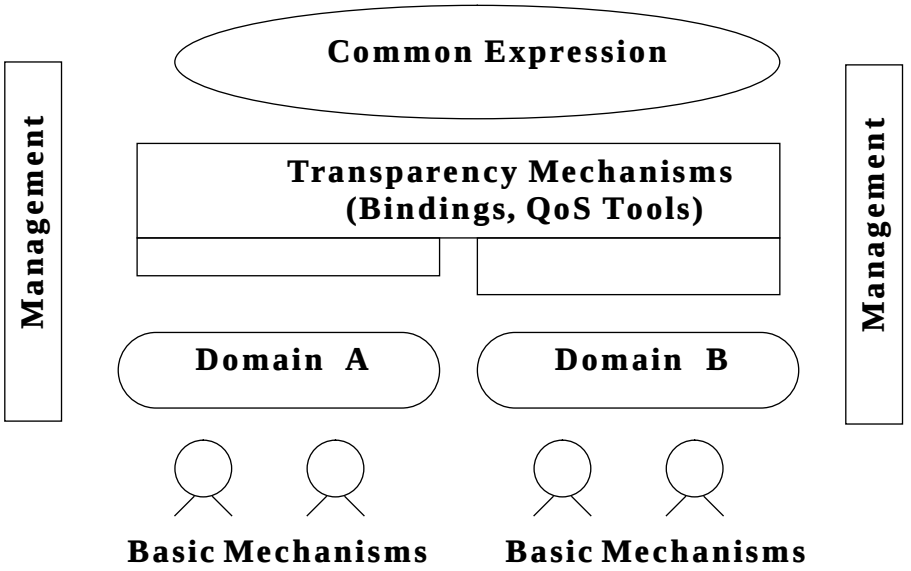
A Language Based Approach to QoS (1)

- **provide a required level of abstraction by the definitions of**
  - **a standard set of forms for QoS primitives (items),**
  - **a standard set of combinators for QoS expression combination,**
  - **a standard set of rules for QoS conformance test,**
  - **a standard set of rules for QoS negotiation,**
  - **a standard set of QoS domains for common applications,**
  - **a standard set of rules for the construction of new QoS domains out of others.**

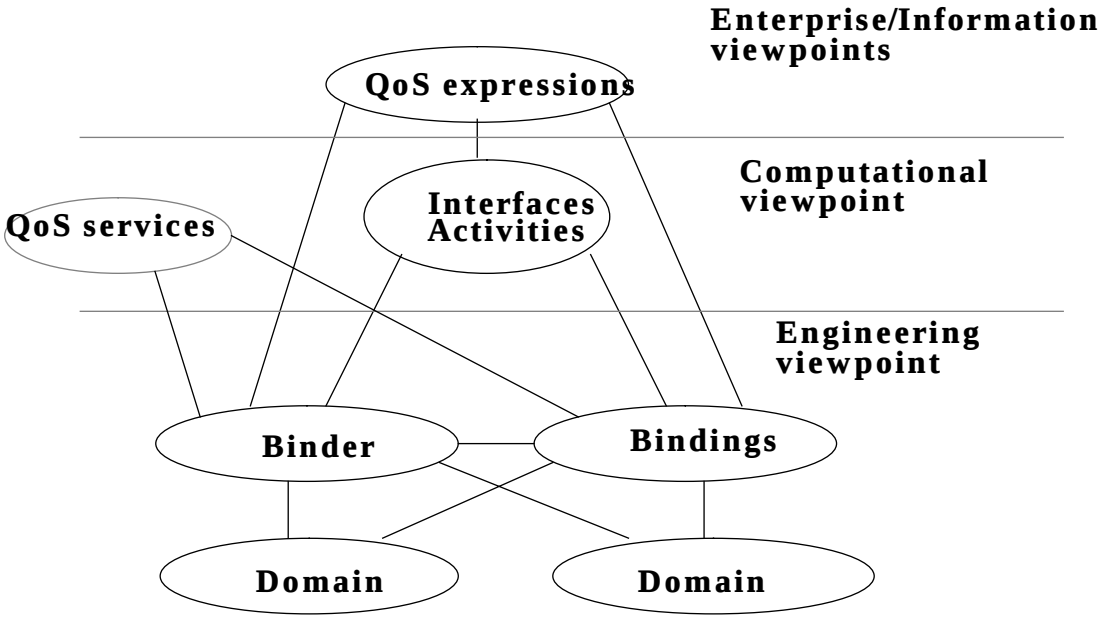
A Language Based Approach to QoS (2)

- **allows the application of automation tools to**
  - **compile application programs onto any suitable technology,**
  - **optimize application programs for any particular configuration,**
  - **transform declarative requirements into imperative statements,**
  - **check that the application program will execute correctly,**
  - **apply the conformance rules,**
  - **apply the negotiation rules.**

Towards a QoS Framework



Viewpoints


Enterprise/Information viewpoints
QoS expressions
Computational viewpoint
Interfaces Activities
QoS services
Engineering viewpoint
Binder
Bindings
Domain
Domain

QoS and Computational/Engineering Entities

- QoS (as templates) can be associated with
  - an interface type
  - an interface
  - an object
  - an invocation or activity
- QoS templates can be used for
  - trading
  - binding
- QoS are implemented as *contracts* on (engineering) bindings

Designing a QoS Language (requirements)

- identify relevant QoS domain,
- identify relevant attributes in a QoS domain,
- express quantitative measures for the identified attributes,
- describe the combination of individual attributes,
- provide conformance check,
- provide negotiation,
- provide the construction of new QoS domains out of existing domains,
- describe domain dependent management/control interfaces.

Designing a QoS Language (syntactic/semantic rules)

- **QoS items**
  - **attributes (keywords) and their types:** **jitter, delay, priority**
- **combinators**
  - **logical combinators:** **QoS1 *or* QoS2**
  - **ordering combinators:** ***prefer*** **QoS1 *to* QoS2**
  - **range combinators:** **(rate >= 100 *and* rate <= 1000)**
- **conformance check**
  - **syntactic analyser, type checker**
  - **semantic --- negotiation**
- **construction**
  - **set, record, union etc.**
- **description of management/control interface**
  - **using the same computational concepts**

A Simple QoS Language

- **the Trader property constraint language**
  - **superlative functions:** **min, max**
  - **comparative functions:** **==, !=, >, >=, <, <=, in**
  - **constructors:** **and, or, not, -> (is restricted by)**
  - **property names:** **identifiers**
  - **numeric and string constants:** **100, “TCP”**
  - **mathematical operators:** **+, -, *, /**
  - **well defined matching criteria:** **semantics**
- **a QoS domain definition language (an extended IDL)**
  - **QoS items**
  - **QoS domain as type**
  - **QoS domain constructor**
  - **domain dependent management/control interface**

QoS Domain and Expression Examples

```
MEDIA:  QoSDOMAIN =
NEEDS CHANMEL;

BEGIN
 Type: KEYWORD = {Audio, Video};
 Rate: KEYWORD = INTEGER;
 Delay: KEYWORD = INTEGER;
 Encoding: KEYWORD = {Pal, Mpeg, Jpeg};
 Channel: KEYWOED = CHANNEL.K;

M_Binding: INTERFACE =
 BEGIN
    Unbind: OPERATION [] RETURNS [];
 END.

 audio(x,y): MACRO "(Rate>=x and Delay<=y)";
 video(x,y,z):MACRO "(Rate>=x and Delay<=y and Encoding==z)";

Bind: OPERATION [ Type: String;
                  EndMgrIf: InterfaceRef;
                  SourceMgrIf: INterfaceRef ];
        RETURNS  [M_Binding, InterfaceRef];
END.
```

```
--QoS expressions on MEDIA
 (Type == Video) and video(100,10,Pal)
```

QoS Matching Example

- **Offers**
  - **(Protocol == TCP) and (Rate == 100)**
  - **(Protocol == TCP) and (Rate == 1000)**
  - **(protocol == UDP)**
- **constraint expression (Protocol == TCP) would return:**
  - **(Protocol == TCP) and (Rate == 100)**
  - **(Protocol == TCP) and (Rate == 1000)**
- **(Protocol == TCP) and (->max[Rate]) would return:**
  - **(Protocol == TCP) and (Rate == 1000)**

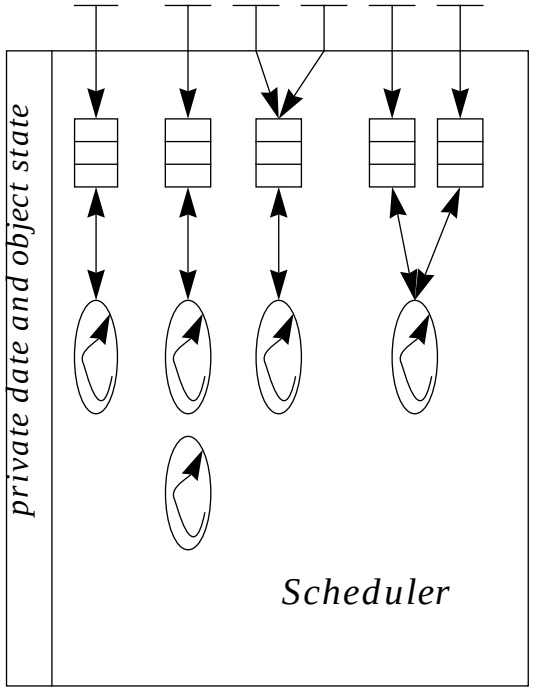
QoS Template Example

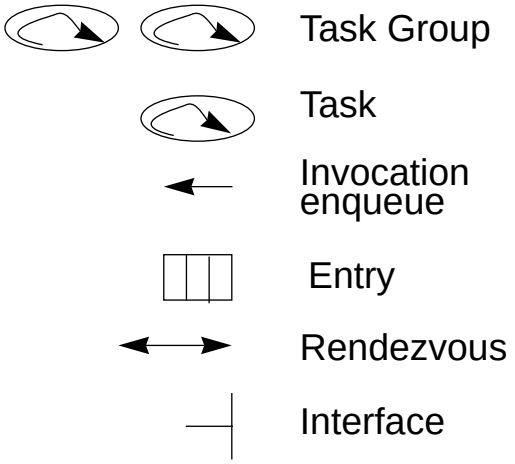
```
VideoDev:   INTERFACE =
NEEDS MEDIA;
  IF_QOS   (Delay <= 100) and (Encoding == pal)
BEGIN
  StreamIn:    OPERATION []  (Rate <= 1000) RETURNS [];
  StreamOut:  OPERATION [] (Rate >= 100) RETURNS [];
END.
```

A Model of Real-Time Programming

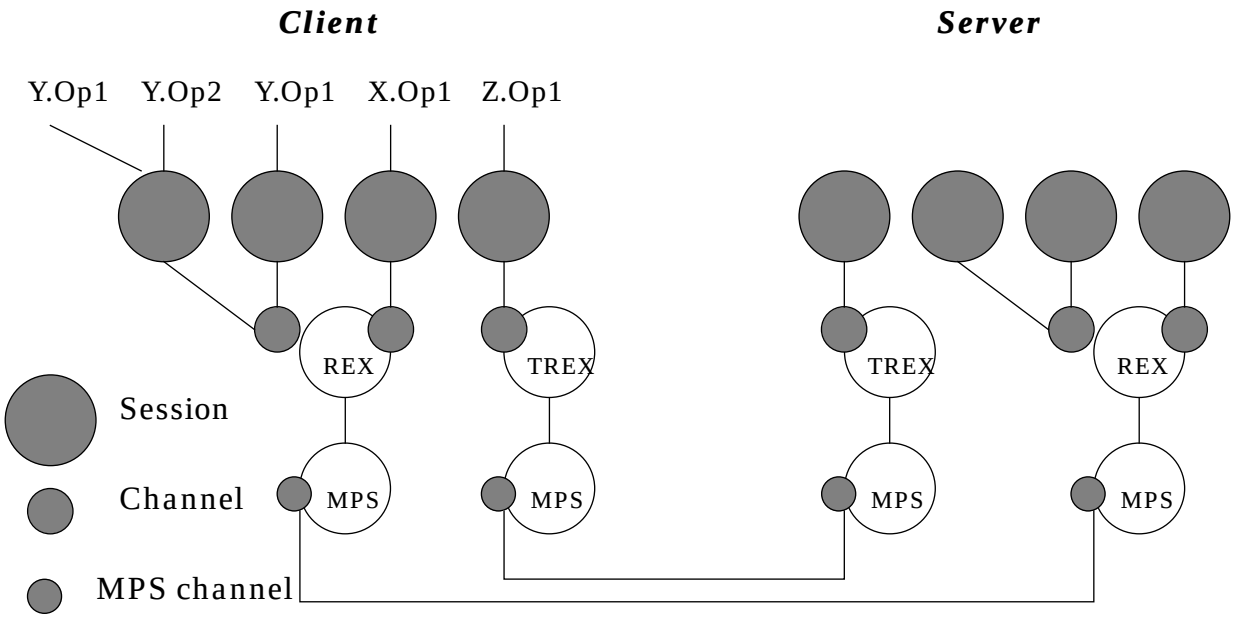
- RC 1072: Engineering Aspects of Real-Time
- a real-time tasking model (real-time objects)
  - real-time invocations: can associate a priority and/or a deadline
  - resource management
  - real-time tasks
  - real-time scheduling
  - priority-based real-time programming model
  - deadline-based real-time programming model
- a real-time communication model
  - parallel protocol stack to allow the association of communication QoS
  - a dependable timed RPC protocol --- define the meaning of deadlines
  - an integrated and decomposable RPC protocol stack --- for supporting different transportation QoS

Real-Time Objects





Real-Time Communication



A Model of Real-Time QoS (1)

- **session QoS domain**
  - **session attributes: priority, deadline, deadline type**
  - **allow implicit binding on real-time channels (set up by explicit binding)**
- **tasking QoS domain**
  - **task attributes: type, tasks, period, priority, stack length**
  - **scheduling entry attributes: type, thread enqueue policy, rendezvous policy, concurrency (queue length)**
  - **allow explicit binding**

A Model of Real-Time QoS (2)

- **communication QoS domain**
  - **two layers: Message Passing Service (MPS) and Execution (EX)**
  - **MPS attributes: protocol, address, rate, bandwidth (depends on the individual platform)**
  - **EX attributes: protocol (REX: Remote Execution Protocol, GEX: Group Execution Protocol, TREX: Timed REX, LREX: Light-weight REX), rate, retry**
  - **allow explicit binding**

Task QoS Domain

```
ENTRY: QoSDOMAIN =
BEGIN
  Type: KEYWORD = {shared, private};
  EnqueuePolicy: KEYWORD = {FCFS, PB, DB, PDB, DPB};
  RendezvousPolicy: KEYWORD = {Null, PI, DI, Ceiling, PDI};
  Concurrency: KEYWORD = INTEGER;
END.

TASKING: QoSDOMAIN =
NEEDS ENTRY, TASK;
BEGIN
  Entry: KEYWORD = ENTRY.K;
  Task: KEYWORD = TASK.K;

  T_Binding: INTERFACE =
  BEGIN
    Unbind: OPERATION [] RETURNS [Result];
    Rebind: OPERATION [] RETURNS [Result];
  END.

  Bind: OPERATION [if: InterfaceRef ]
          RETURNS [T_Binding];
END.
```

Summary

- **Major results**
  - **the identification of the requirements for QoS specification**
  - **the identification of the design constraints for QoS**
  - **the development of the required concepts, entities and their structures for a QoS framework**
  - **the application of the framework on real-time**
- **Future work**
  - **a dedicated QoS language**
  - **better understanding of the relation between QoS and Binding**
  - **QoS tools and supporting services**