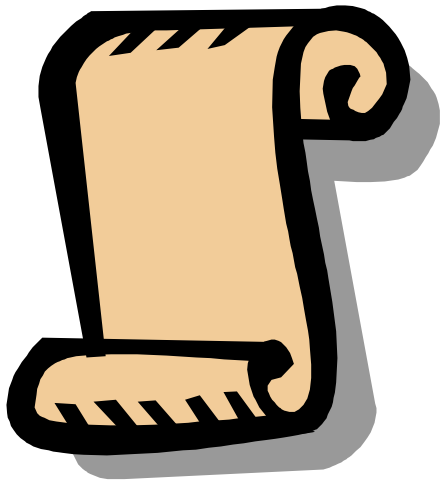
QoS in Middleware

Gordon S. Blair

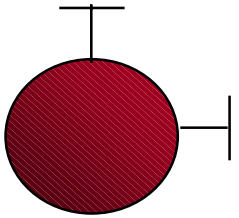
E-mail: gordon@comp.lancs.ac.uk

Overview of the Presentation

- QoS in RM-ODP - an architectural framework
- Experiences of applying this framework
  - The Sumo project
  - The Adapt project
  - The Open-ORB Project
- ‘Reflections’ on this work



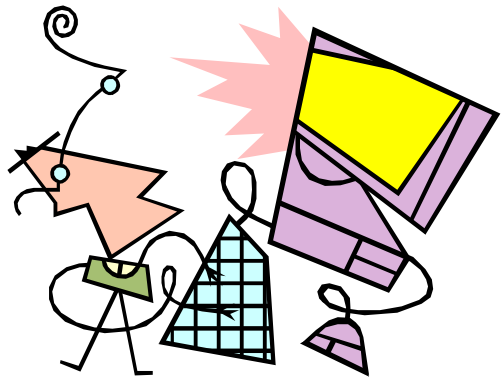
Fundamentals of RM-ODP



- An object-oriented approach
  - » Well defined object model featuring objects, interfaces, templates, types, class, composite objects, ….
- The concept of viewpoints
  - » Viewpoints as a handle on complexity
  - » Five viewpoints defined
    - – Enterprise, Information, Computational, Engineering, Technology

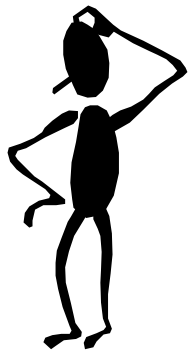
Focus on the Computational Language

- Basic object model
  - » Objects can have multiple (strongly typed) including operational and stream interfaces
  - » Associated environmental contract
  - » Explicit binding supported
- Associated services
  - » Trading
  - » Factory objects

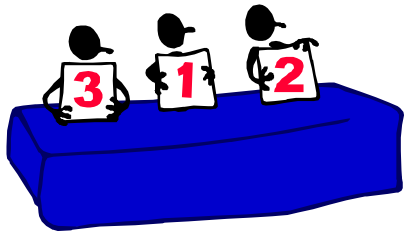


QoS in RM-ODP: Requirements

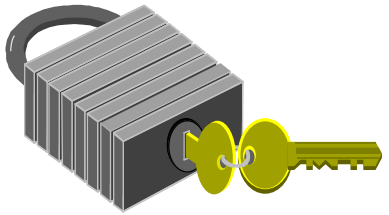
- The static to dynamic spectrum
  - » Static systems fix QoS at design time
  - » Dynamic systems require support throughout lifetime
- End-to-end requirement
  - » Cooperation between all components
  - » Pervasive nature of QoS



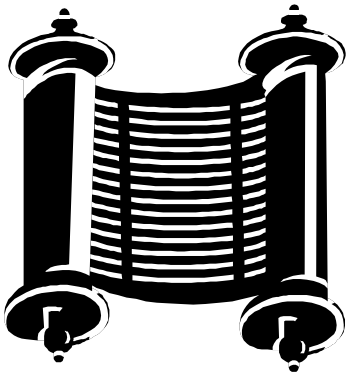
QoS Characteristics



- Time-related
  - » Date/time, time delay, latency, etc
- Capacity-related (cf volume)
  - » Throughput, processor load, etc
- Security-related characteristics
  - » Protection, access control, authentication, confidentiality, etc
- And so on ...



A Contractual Approach



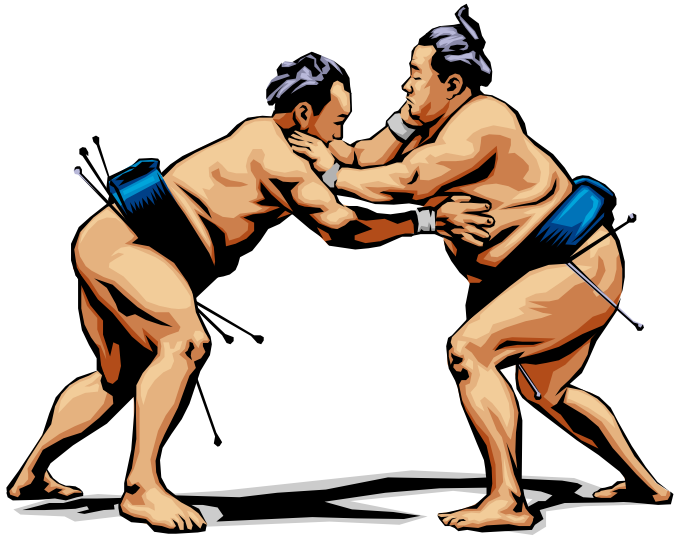
- QoS contracts
  - » Contracts which define the QoS capabilities that objects have agreed to provide to one another and also the requirements they intend to satisfy
- QoS relations
  - » Defined in terms of expectations (on the environment) and obligations (to the environment)

QoS Functions

- Design time
  - » QoS specification
- Instantiation time
  - » Negotiation, refinement, admission testing, resource reservation
- Run-time
  - » QoS monitoring and control

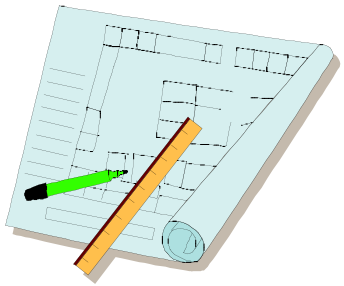
Experiences #1: The Sumo Project

- Aims of the research
  - » To develop an approach to supporting multimedia in the context of RM-ODP
  - » To focus particularly on issues of QoS management

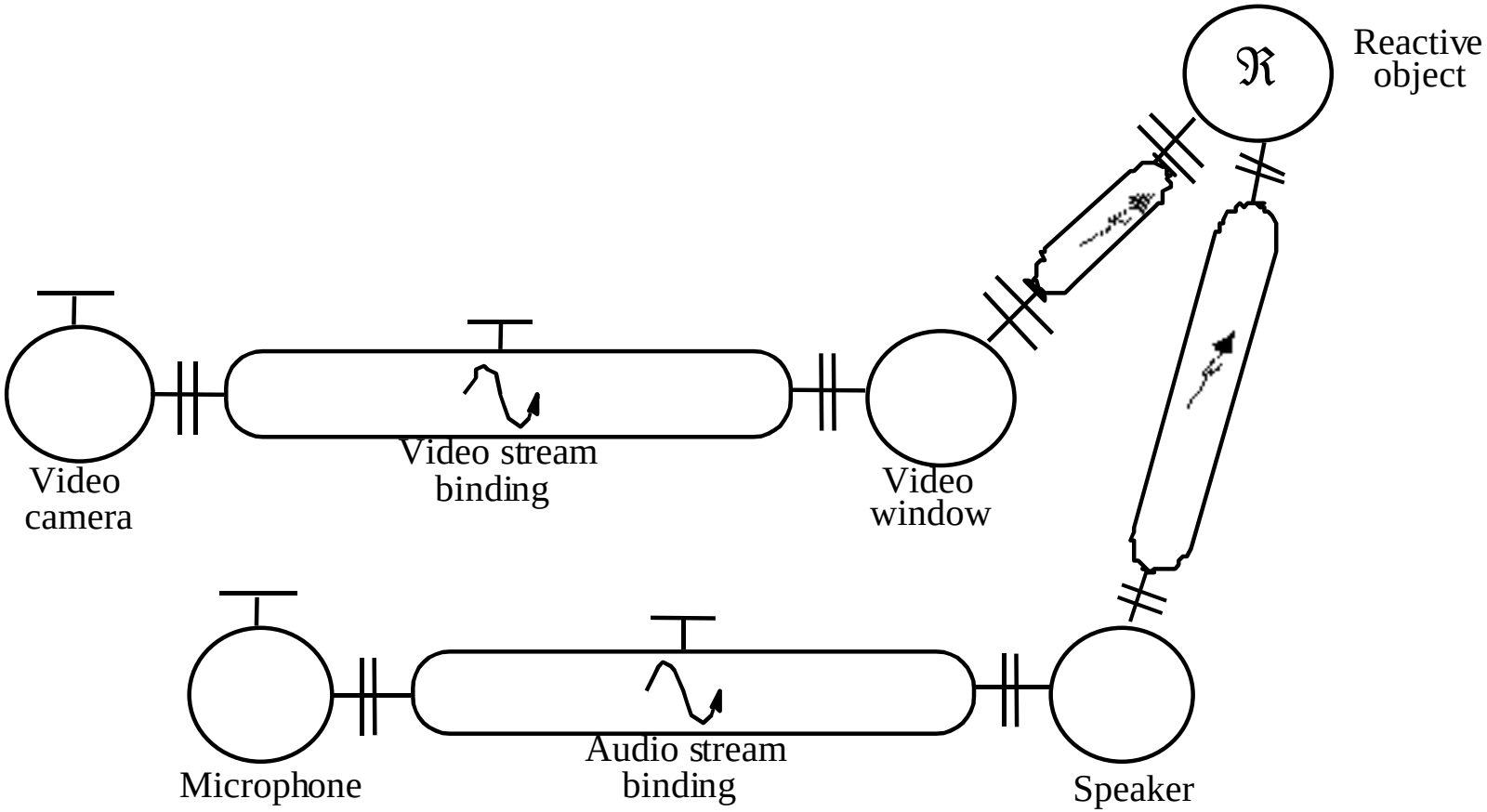


General Approach

- At design time time
  - » QoS annotations on interfaces (contracts) using extended IDL
- At instantiation time
  - » Factories (including binding factories) support range of QoS functions
- At run-time
  - » Reactive objects (written using Esterel) provide monitoring and control



Example: Lip Synchronisation



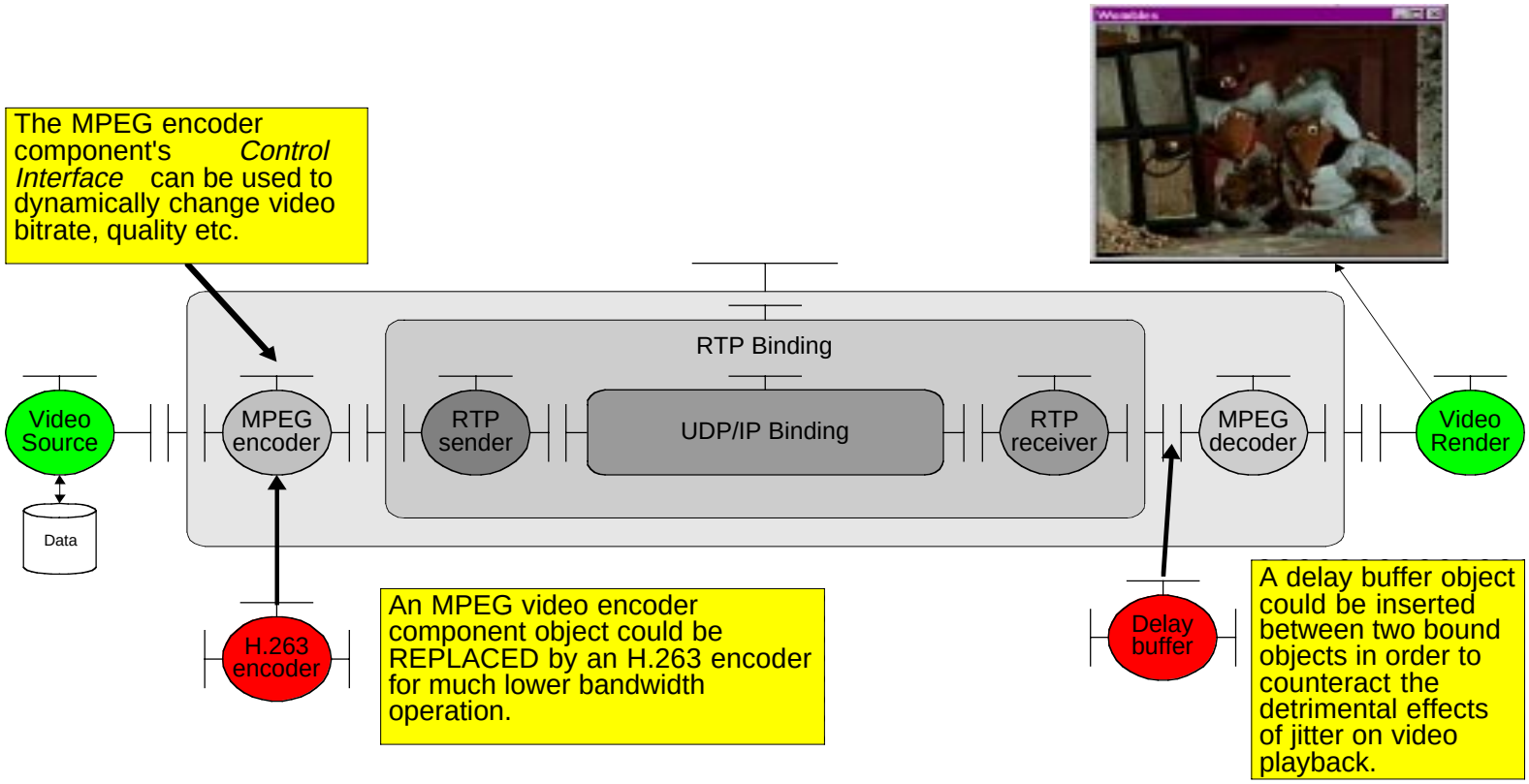
Analysis

- Validation of RM-ODP concepts for multimedia applications
  - » Cf CORBA, DCOM
- Strong endorsement of QoS facilities in RM-ODP
  - » Esp wrt Computational Viewpoint
- Difficulties wrt QoS specification
  - » An extended IDL?

Experiences #2: The Adapt Project

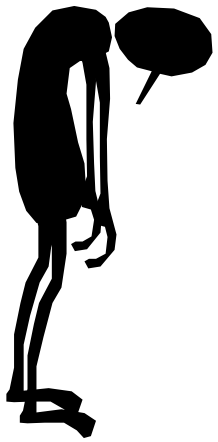
"The main aim of the Adapt Project is
to investigate the development of distributed systems support to manage the differing levels of connectivity a mobile user will experience. More specifically, the Adapt Project will investigate the required support for adaptive multimedia applications which are capable of intelligently adapting to QoS fluctuations (e.g. through QoS re-negotiation, media scaling, etc)."

Example: An Open Binding



Analysis

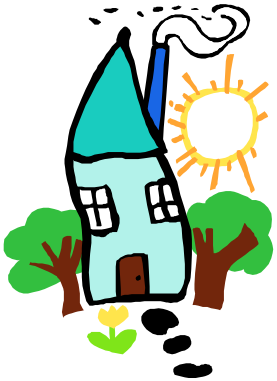
- Towards a more procedural approach to QoS
  - » Inherently more flexible in terms of strategy
  - » Declarative interfaces can be offered on top
- Importance of open implementation
  - » Supporting introspection and adaptation
- Scope too limited
  - » Open implementation limited to bindings



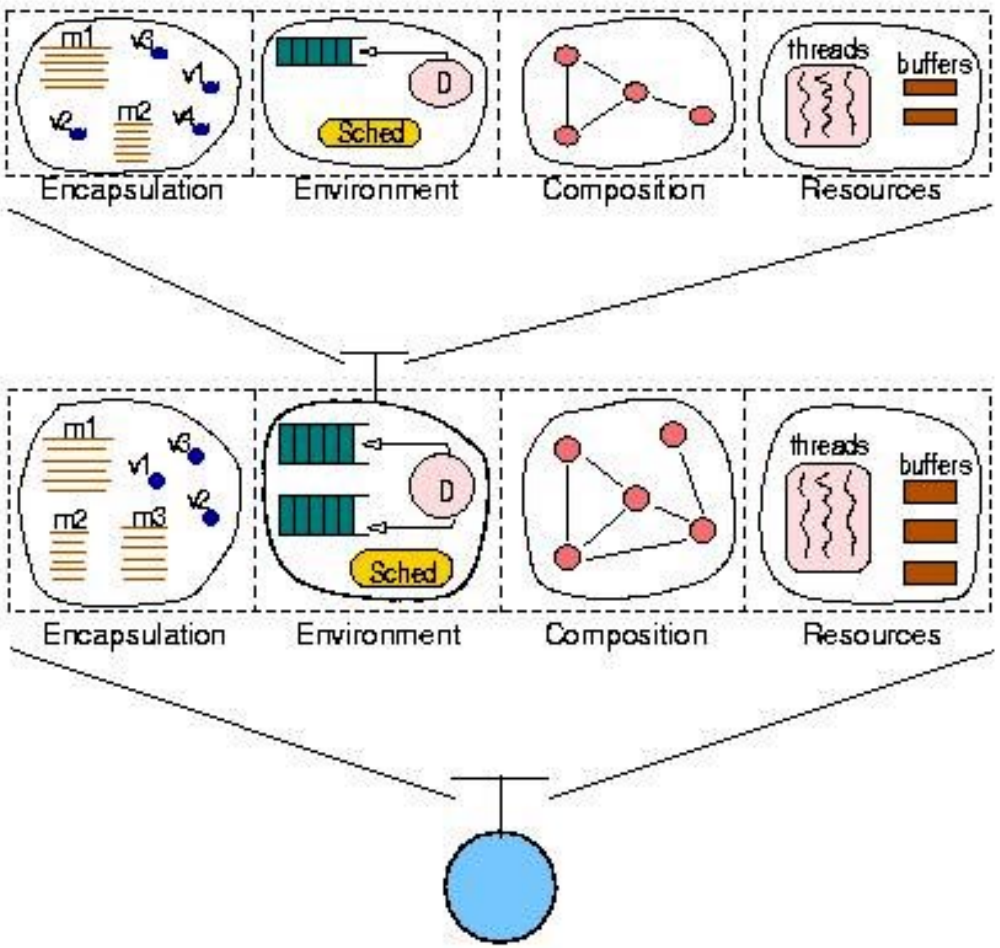
Experiences #3: The Open-ORB Project

- Aims of the research
  - To implement a more complete open middleware platform based on the concept of *reflection*
  - Long term goal of specifying a CORBA MOP

- Initial prototype
  - Constructed using reflective Python
  - Featuring a number of distinct meta-models

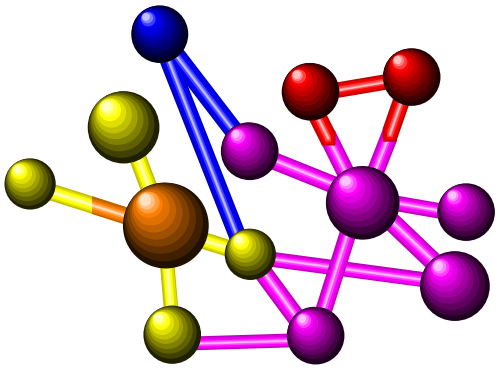


The Open ORB Architecture

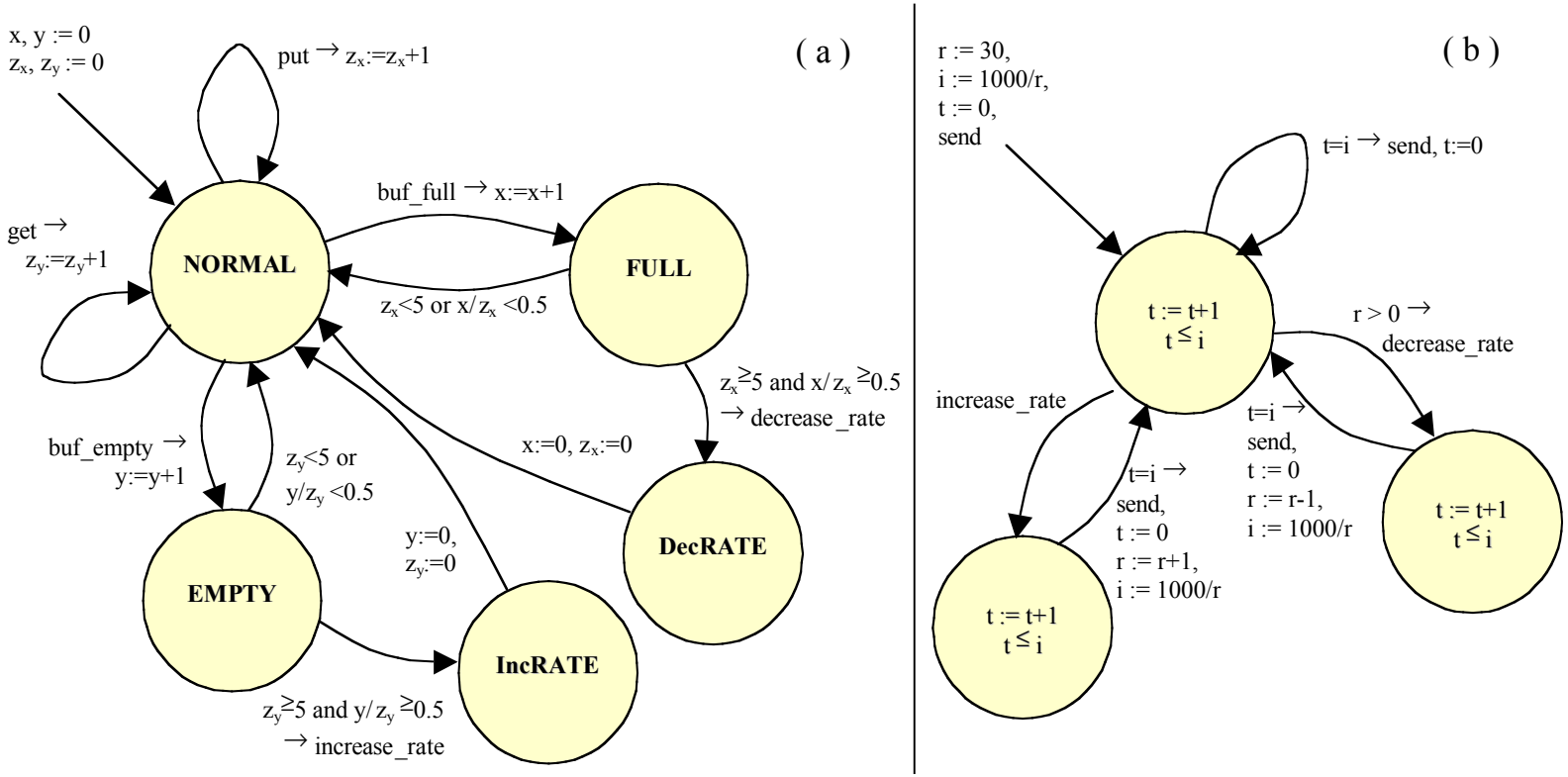


Introducing QoS Management

- Overall approach
  - » Inject QoS management components into meta-space
  - » Interaction via event notification service
- Styles of management component
  - » Monitors
  - » Controllers
    - – Strategy selectors
    - – Strategy activators



Specifying Policies: The Use of Timed Automata



…. and then there are meta-policies!

Analysis

- A principled approach to QoS management
  - » Esp. introspection and adaptation
  - » Generally more dynamic and flexible
    - – e.g. exploiting meta-meta-levels
- A more convincing interpretation of viewpoints
  - » I.e. wrt the underlying platform

Overall ‘Reflections’

- QoS management is crucial area for the design of next generation middleware
- QoS management implicitly requires open access to underlying system functionality
- Reflection provides a principled means of achieving this open access