# A reflective component-based approach to building Grid systems

Geoff Coulson

*Distributed Multimedia Research Group,*
*Department of Computing,*
*Lancaster University*

geoff@comp.lancs.ac.uk

1

## Overview

- our previous work on reflective middleware
  - *components, reflection, component frameworks*
- using our approach to address two issues in Grid computing
  - collaborative distributed vizualisation: current services are monolithic and lack distributed systems support => *component-based Grid vizualisation*
  - OGSA: a more principled middleware approach to Grid computing, but has key limitations => *"extensible binding types" and fine-grain resource management for OGSA and .NET*
- (future focus on *flexible communications services* via "open overlays")

2

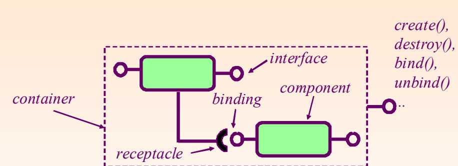
# Our approach to systems building

1. uniform component model (“*components everywhere*”)
   - fine-grain, language- and platform-independent component-based programming model
   - used for both ‘middleware’ and ‘applications’
2. reflective meta-models (*flexibility*, *openness*)
   - provide fundamental support for configuration, reconfiguration, and system evolution
   - built into the component model
3. component frameworks (*coarse-grain structure*, *constraint*)

3

# OpenCOM: reflective components

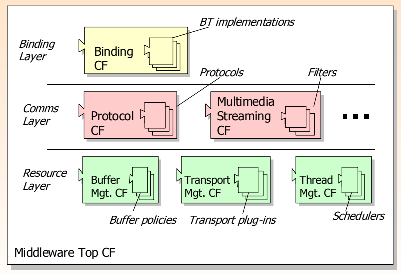
- derived from a core subset of MS COM but now entirely independent of this
- four orthogonal reflective meta-models
  - architecture
  - interface
  - interception
  - resources




4

# Component frameworks

- domain-specific 'life support environments' for plug-in components
- define *roles* and *constraints*
- e.g., OpenORB: an OpenCOM/CF-based reflective middleware platform
  - highly configurable and reconfigurable
  - constraint example: streaming CF => real-time threads




5

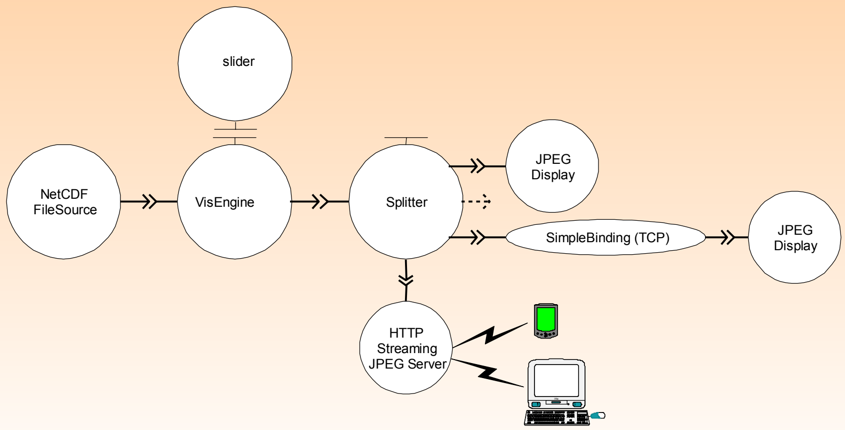
# Issue 1: limitations of current scientific vizualisation services

- AVS/Express, Iris Explorer, ViSAD, etc. don't really support *collaborative* distributed vizualisation
- extensions (e.g. Manicoral, COVISE) are typically:
  - monolithic
    - hard to extract functionality (e.g. file readers, viewers, transformers, splitters, ...), spread it sround the network, and combine it in unanticipated ways
    - running per-user identical copies of the visualization software limits adaptation to heterogeneous environments
  - lack fundamental distributed systems support
    - e.g. migration, resource management, fault-tolerance, persistence, load balancing, logging, ...

=> *open, extensible, component-based platforms*

6

## Approach: component-based vizualisation in "Visual Beans"




7

## Issue 2: limitations of OGSA

- WSDL describes services; SOAP provides messaging, request-reply interactions, and intermediaries
- advanced applications require more than this
  - different modes of interaction – e.g. multicast, streaming, pub-sub, tuple spaces, ...
    - extensible set – and QoS management to underpin them
  - also
    - need for internal platform *architecture*
      - to integrate horizontal and vertical system elements
      - for building *self-managing* systems

=> *open, extensible, component-based OGSA*

8

## Approach: “extensible binding types” for Grid middleware

- already designed a CF for this in OpenORB
- examples of “binding types”
  - RMI + variants, messaging and eventing (async. invocation, message queuing, pub/sub), media streaming, group comms + variants, shared data spaces (tuple spaces, blackboard systems, mailboxes), SQL or FTP links, voting and auction protocols, distributed resource allocation protocols, workflow processes, multi-player game protocols, ...
- now adding this to OGSA and .NET (2 PhD projects)
- also adding *resources meta-model* to underpin QoS provision with fine-grain resource management
  - intention to integrate with coarse-grain protocols like GRAM, SNAP

9

## Future work: “open overlays:” Grid-oriented network abstractions

- goals
  - facilitate the building of highly configurable ‘virtual networks’
    - e.g., with ringfenced resources, migration of virtual routers
  - decentralised, autonomic, (e.g. gossip protocol, or ‘virtual fireflys’)
  - integrate with component-based middleware and apps – and programmable networks (fixed, wireless, ubicomp)
- our usual CF-based approach
  - define (distributed) component frameworks
  - instantiate in terms of plug-in components
    - e.g. roles and constraints for routers, resource managers, proxies, ...

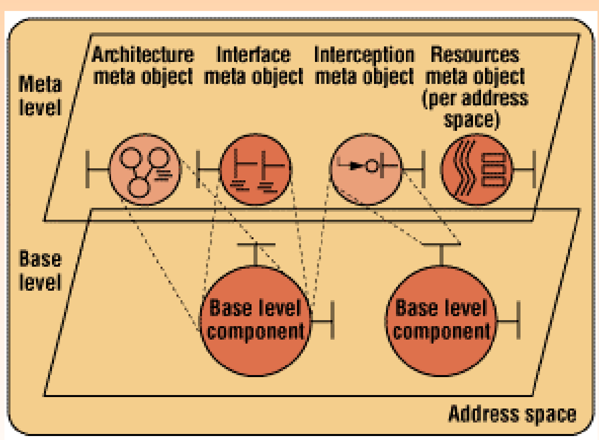
10

# Conclusions

- we are investigating a reflective component-based approach to Grid middleware
  - uniform programming model (components everywhere)
  - reflective meta-models (flexibility, openness)
  - component frameworks (structure, constraint)
- approach already validated in ORB environment
- now applying to Grid-oriented services
  - study of collaborative vizualisation
  - “extensible binding types” for Grid platforms
- future work
  - “open overlays” for flexible communications support

11

# Four reflective meta-models




12

## Stuff

- co-locating/ separating visualization and other application processing
- other limitations of OGSA...
  - little architectural framework
    - need to support integration of diverse system elements
      - *breadth*--generic distributed services like persistence, visualization
      - *depth*--services in intimate contact with networks/ systems
  - no support for complexity management
    - scale and complexity demands *self*-managing approach.
    - self-management of whole system, including comms services

13

## Scientific vizualisation issues

- distributed scientific vizualisation
  - exposes interesting resource allocation issues relating to optimal placement of processing and display, e.g.
    - moving processes to data or vice versa
    - computational steering
- *collaborative* visualization
  - multiple scientists sharing related data sources
  - even richer set of architectural and resource management issues
    - sessions evolve over time and involve heterogeneous participants
    - optimal resourcing changes as members join/leave and as data sources are added/removed or fall in/fall out of scope
    - possible user mobility

14