Managed by

The Grid Component Model: an Overview

"Proposal for a Grid Component Model" DPM02

"Basic Features of the Grid Component Model (assessed)" -- DPM04

CoreGrid institute on Programming Models

Ludovic Henrio

Context

- “By defining the GCM, the institute aims at the precise specification of an effective Grid Component Model.”
- The features are discussed taking Fractal as the reference model, and defined as extensions to the Fractal specification (also relates to CCA, CCM, …)
- The institute expects several different implementations of the GCM, not necessarily relying on existing Fractal implementations.

General Features

- Component hierarchy
- Extensibility of the model
- Support for adaptivity
- Language neutrality
- Interoperability
- Reflexivity

➢ Lightweight ⎝ portable and compact implementations

➢ Well-defined semantics (allow future formalization)

Outline

- A Short Summary of Fractal
- Abstract Model of the GCM
- Communication, Parallelism and Distribution
- Dynamic Controllers
- Support for Autonomicity

GCM is Based on Fractal

Fractal provides:

- ***Terminology*, *API* (and *ADL)*** ⎝ Interoperability
- ***Hierarchical* structure**
- ***Separation of concerns***

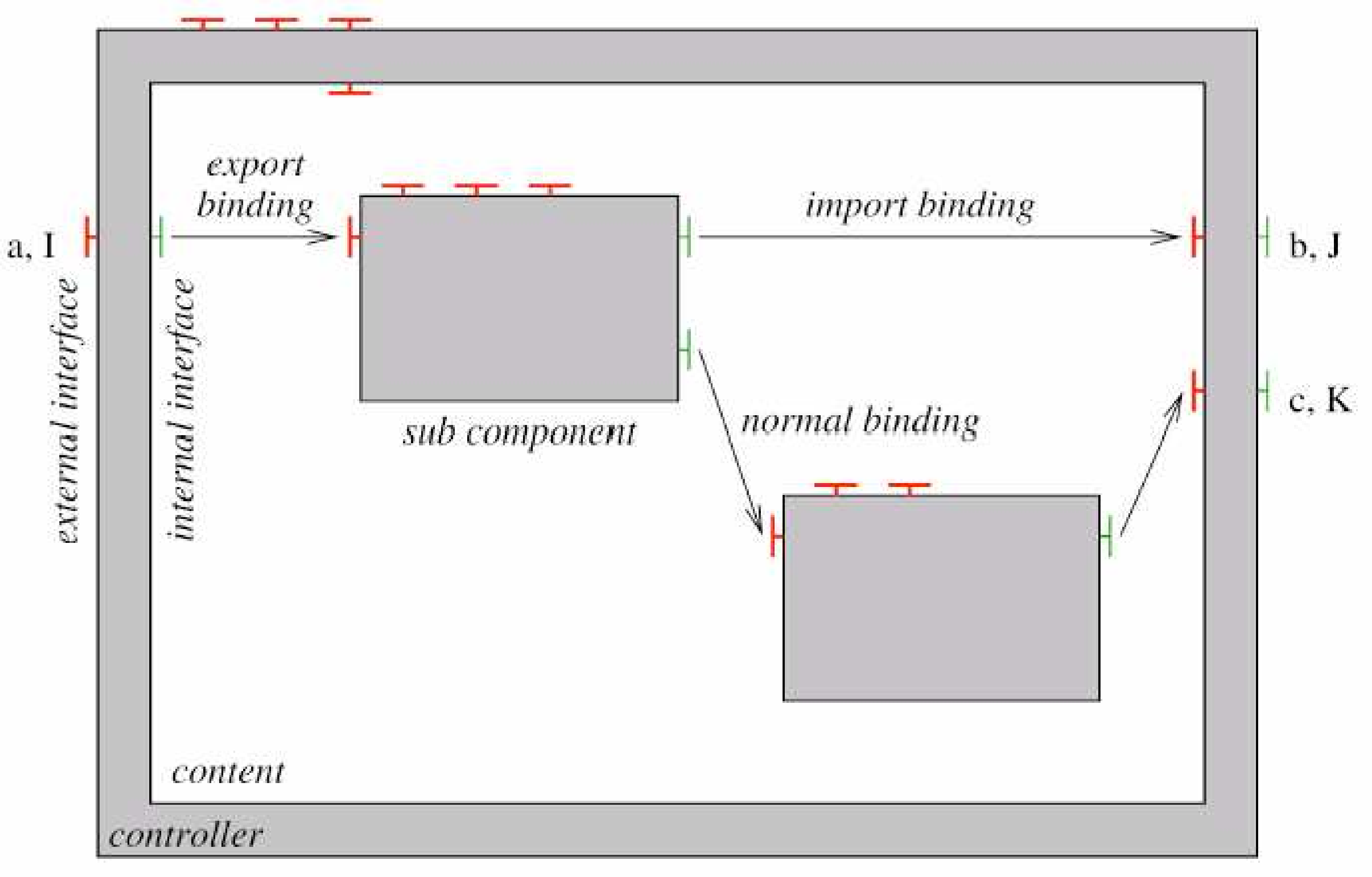
general features

- ***Abstract* component model** ⎝ **no constrain on implementation:** several implementations exist
- ***Multi-level* specification: almost every object is a level 0 Fractal component**

Multi-level specification of the GCM

⎝ We focus on the Grid specific extensions of Fractal

A Fractal Component



Outline

- A Short Summary of Fractal
- Abstract Model of the GCM
- Communication, Parallelism and Distribution
- Dynamic Controllers
- Support for Autonomicity

Defining and Deploying Components

- Under standardization
- XML Component Specification

  (XML schema or DTD)
- Run-Time API defined in several languages
- Packaging described as an XML document

  cf. Fractal packaging

Definition / Description of a Component

- Definition of Primitive Components
- Definition of Composite Components (composition)
- Interfaces (ports) : Server, Client / asynchronous method calls, event, stream, etc.
- Specification of Grid aspects:
  - Parallelism, Distribution, Virtual Nodes,
  - Performance Needs, QoS, etc.
- Including external references to various specifications:
  - Java Interface, C++ .h, Corba IDL, WSDL, etc.

Outline

- A Short Summary of Fractal
- Abstract Model of the GCM
- Communication, Parallelism and Distribution
- Dynamic Controllers
- Support for Autonomicity

Communications

- Semantics should be specified in the interfaces
- asynchronous method call is the default
- Implementation details purposely unspecified
- Allows streaming and event-based communications

Parallel Components: Distribution

- Notion of Virtual Nodes ⎝ distribution
  - Maps the virtual architecture to a physical one
  - One can envisage more sophisticated information such as, for instance, topology information, QoS requirements between the nodes, etc.
- Parallel components can
  - Be distributed or not
  - Admit several implementation ⎝ adaptive implementations

Collective Communications

Fractal type-system

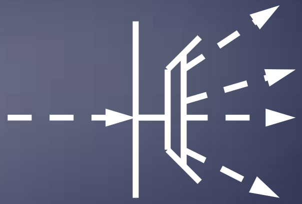
Simple type system

Component type ← types of its interfaces

Interface type :

- Name
- Signature
- Role
- Contingency
- Cardinality **extended to support multicast / gathercast**

Multicast interfaces



Transform a single invocation into a list of invocations

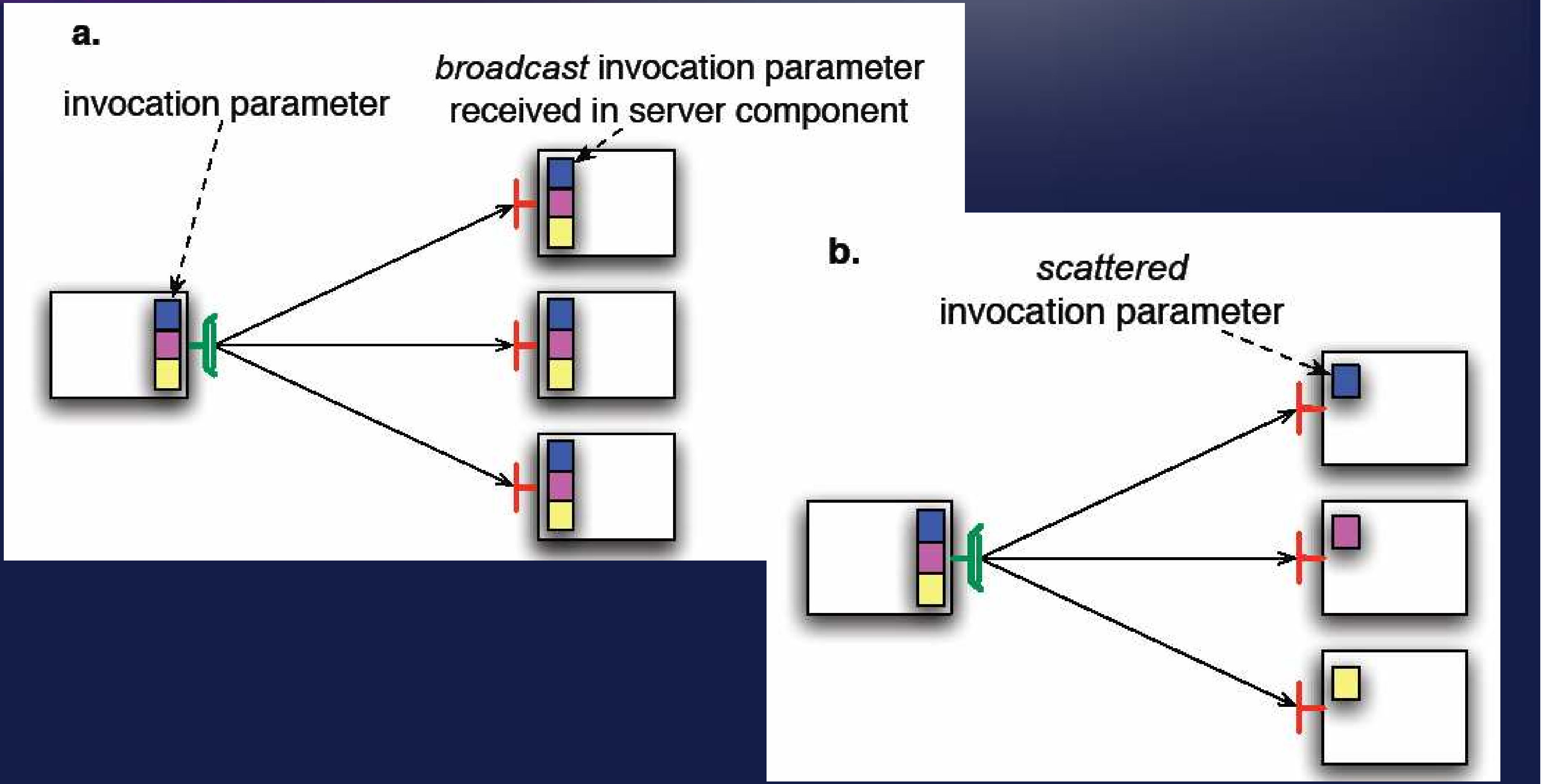
Multiple invocations

- Parallelism
- Asynchronism
- Dispatch

Data redistribution (invocation parameters)

- Parameterisable distribution function
- Broadcast, scattering
- Dynamic redistribution (dynamic dispatch)

Result = list of results



a.

invocation parameter

broadcast invocation parameter received in server component

b.

scattered invocation parameter

Gathercast interfaces

Transform a list of invocations into a single invocation

Synchronization of incoming invocations

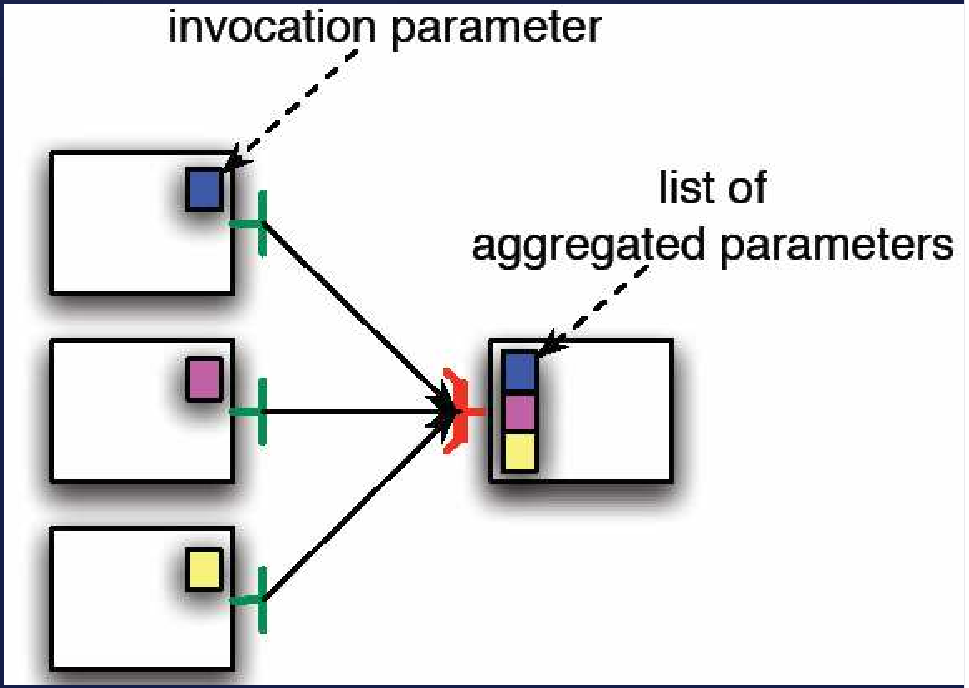
- ~ “join” invocations
- Timeout / drop policy
- Bidirectional bindings (callers ⇔ callee)

Data gathering

Aggregation of parameters into lists

Redistribution of results

Redistribution function



Collective interfaces

- Multicast / Gathercast / Gathermulticast
- Specific API
- Allow MxN communications:
  - Redistribution and direct communications

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

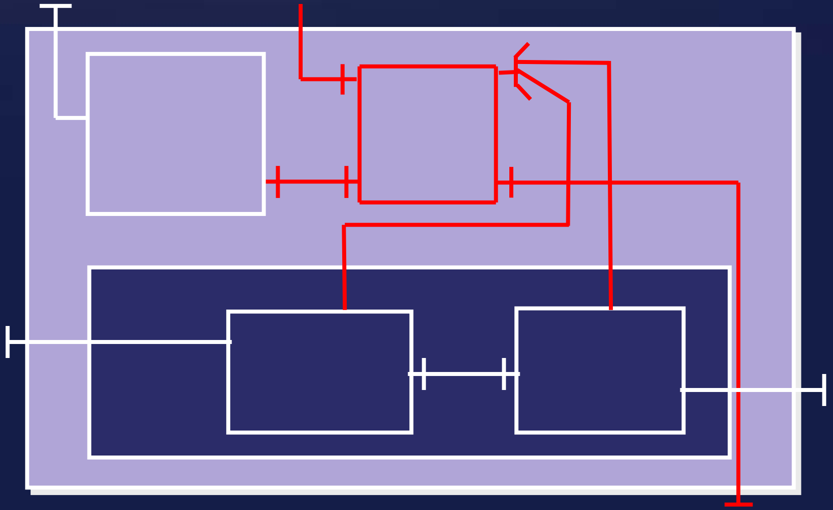
Outline

- A Short Summary of Fractal
- Abstract Model of the GCM
- Communication, Parallelism and Distribution
- Dynamic Controllers
- Support for Autonomicity

Dynamic Controllers
(components in component's membrane)

- Interest for the GCM:
  - Reconfiguration and adaptativity of the membranes
  - For autonomic aspects: hierarchical composition of autonomic aspects (also multicast)

→Fractal (GCM) Components in the membrane



- Apply Fractal specification to the non-functional aspect
  - Pluggable NF server interfaces (NF components)
  - NF client interfaces

Dynamic Controllers Summary

- Adaptativity and autonomicity
- Allow dynamic reconfiguration of the controllers
- Better separation of concerns
- Modification of the content controller (for the membrane)
- Controller components should be *lightweight* components
- Might have restriction on distribution or complexity of component controllers
- Conformance levels: component controllers are optional
- Refinement of the specification and implementation of component controllers

Outline

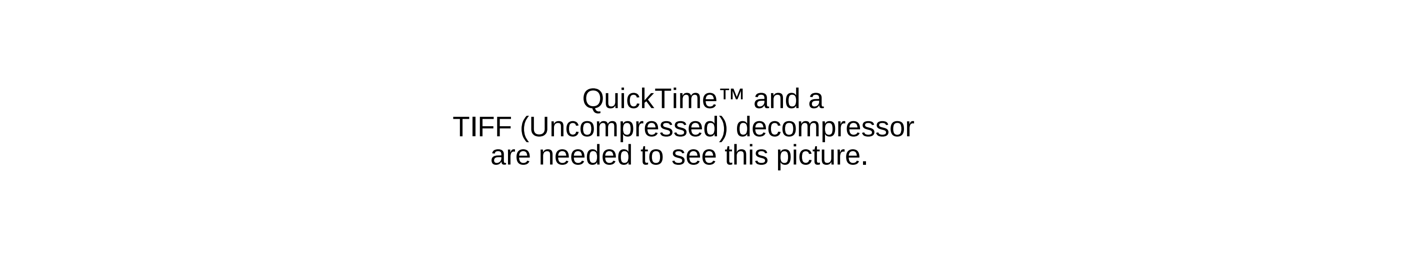
- A Short Summary of Fractal
- Abstract Model of the GCM
- Communication, Parallelism and Distribution
- Dynamic Controllers
- Support for Autonomicity

Autonomicity

- **Self-Configuring**: handles reconfiguration inside itself
- **Self-Healing**: provides its services in spite of failures
- **Self-Optimising**: adapts its configuration and structure in order to achieve the best/required performance.
- **Self-Protecting**: predicts, prevents, detects and identifies attacks, and to protect itself against them.
- Open and extensible specification
- Several levels of autonomicity depending on:
  - autonomic controllers implemented
  - autonomicity level implemented by each controller

Specification of Autonomicity

- Three levels for autonomicity
  - No autonomic control
  - Passive autonomic control
  - Active autonomic control
- API:



QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

- Should **compose with hierarchy** and might use **component controllers**

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Summary / Conclusion

- Hierarchical and extensible component model
- Support for distribution and extended communication patterns
- Multicast/Gathercast specification + implementation: collective communications

« Component Oriented SPMD »

- Deployment of components
- Dynamic controllers
- Autonomicity (passive / active)

Current and Future Works

- MxN as an optimization for the coupling of multicast and gathercast (Elton)
- Generalisation of the data distribution and gathering policies for multicast and gathercast (new conditions on typing)
- Reconfiguration primitives adapted to distribution
- Experiments and validation

➔ a prototype implementation of the GCM under GridCOMP: ProActive/GCM

Questions?

Summary: Requirements and Concepts

Hierarchical composition → Fractal

Extensibility → From Fractal design

→ dynamic controllers (for non-functional)

→ open and extensible communication mechanisms

Support for reflection → Fractal specification and API

Lightweight → Conformance levels

→ No controller imposed

ADL with support for deployment → Virtual Nodes

Packaging → packaging being defined by the Fractal community

Support for deployment → Notion of virtual nodes

→ ADL with support for deployment

Hierarchical and parallel implementation → XML component specifications, and Multicast-Gathercast interfaces allow plugging and unplugging several componentsto the same interface dynamically

Asynchronous ports and Extended/Extensible port semantics

→ Asynchronous Method Invocation as default but can be defined via tags; + Possibility to support method calls / message oriented / streaming / …

Group related communication on interfaces

→ Multicast / Gathercast interfaces

Interoperability → Exportation and importation as web-services

Language neutrality → API in various languages

→ Various interface specifications

→ exportation of a web-service port

→ Globally due to dynamic controllers

- Exploit Component Hierarchical abstraction for adaptivity

  → Dynamic controllers
- plug/unplug component → Fractal: content + binding controller
- Give a standard for adaptive behavior and unanticipated extension of the model → Dynamic controllers
- Give a standard for the autonomic management components

  → Autonomic controllers
- Plug/unplug non-functional interfaces → Dynamic controllers

Parallel binding: Well-defined and verifiable composition

→ Multicast / Gathercast

Conclusion

Future (technical) works: model has to be refined

- Technical issues (dynamic controllers)
- APIs
- extended ADL (behaviour, dynamic controllers)
- …

Like in Fractal, we aim at a multi-level specification, → an implementation of the GCM can be level 1.1 Fractal compliant and level 1.2.1 GCM compliant. GCM levels to be specified

Next steps: assessment, experiments, (reference) implementations (→ GridCOMP)

Questions / Comments ?