Components for the Grid with ProActive and Fractal

Matthieu Morel

OASIS Team - INRIA

ProActive User Group

October 10th 2005

Rationale

- ❑ Observation : complexity and heterogeneity of the Grid
  - ➡ complexity (design, deployment and reusability)
  - ➡ performance issues

- ❑ Answer : framework for programming and deploying components on the Grid
  - ➡ implementation of the Fractal model for ProActive
  - ➡ extensions for the Grid

Objective :
a framework for Grid components

- ❑ Facilitating the design and implementation of complex distributed systems
- ❑ Leveraging the ProActive library
  - ProActive components benefit from underlying features
- ❑ Allowing reuse of legacy components (e.g. MPI)
- ❑ Providing tools for defining, assembling and monitoring distributed components

Agenda

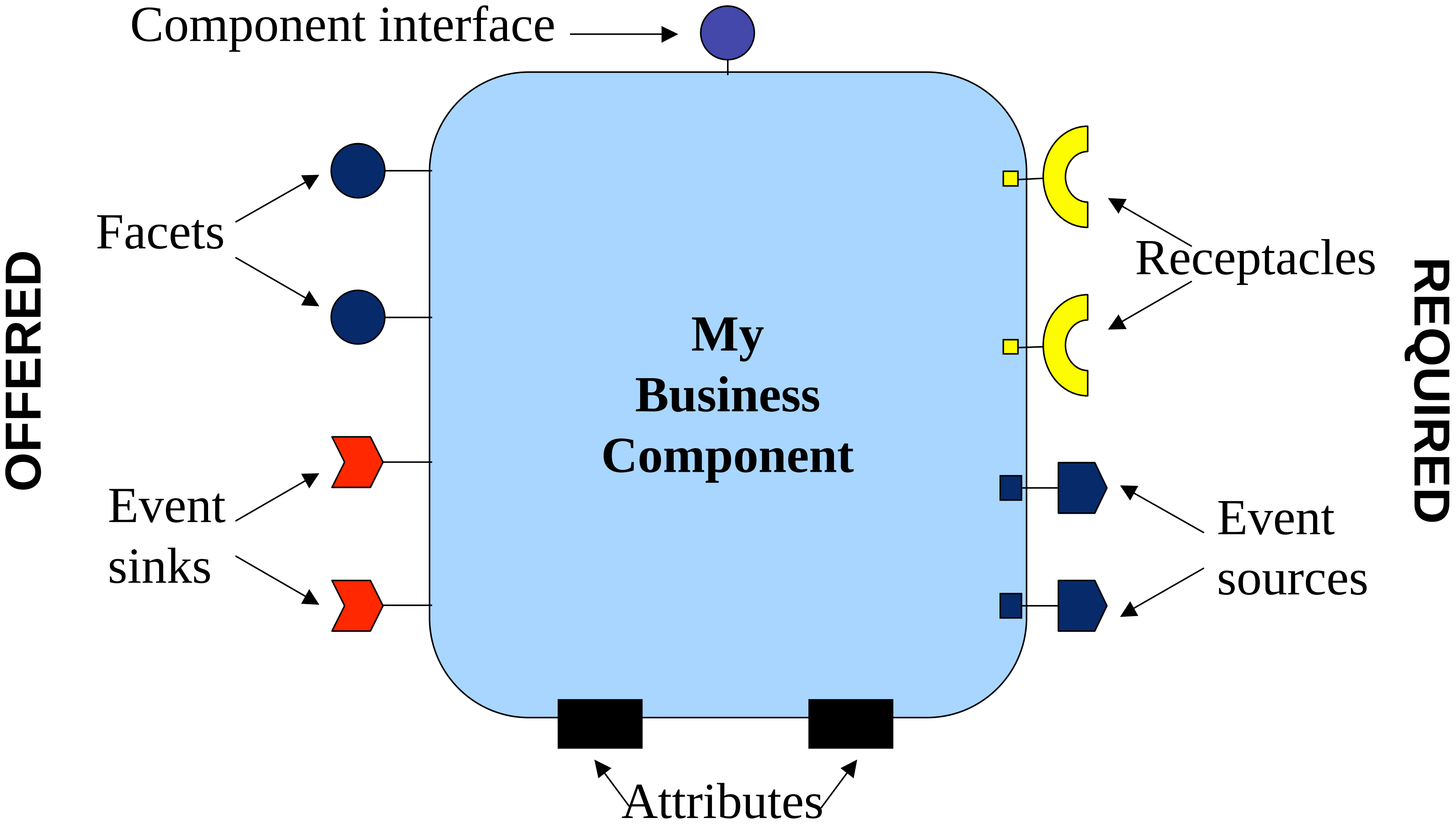
- ❑ Component-based programming
- ❑ Fractal component model
- ❑ Components for Grid computing
- ❑ On-going work

Componentizing software

- ❑ *Douglas McIlroy :*

  Mass Produced Software Components, 1968

A CORBA Component



Component based programming

- Component = software unit, deployment unit
- Industrial acceptance : EJBs, CCM, COM ...
- 3 key concepts :
  - 1. Encapsulation
    - Black boxes, offered and required services, configuration
  - 2. Composition
    - Design of complex systems
    - Hierarchical organization into sub-systems
    - Replacement
  - 3. Description
    - ADL, QoS
    - Logical and geographical composition
    - Tools

HIGH ABSTRACTION LEVEL

COMPLEXITY HANDLING

REUSABILITY

CUSTOMIZATION

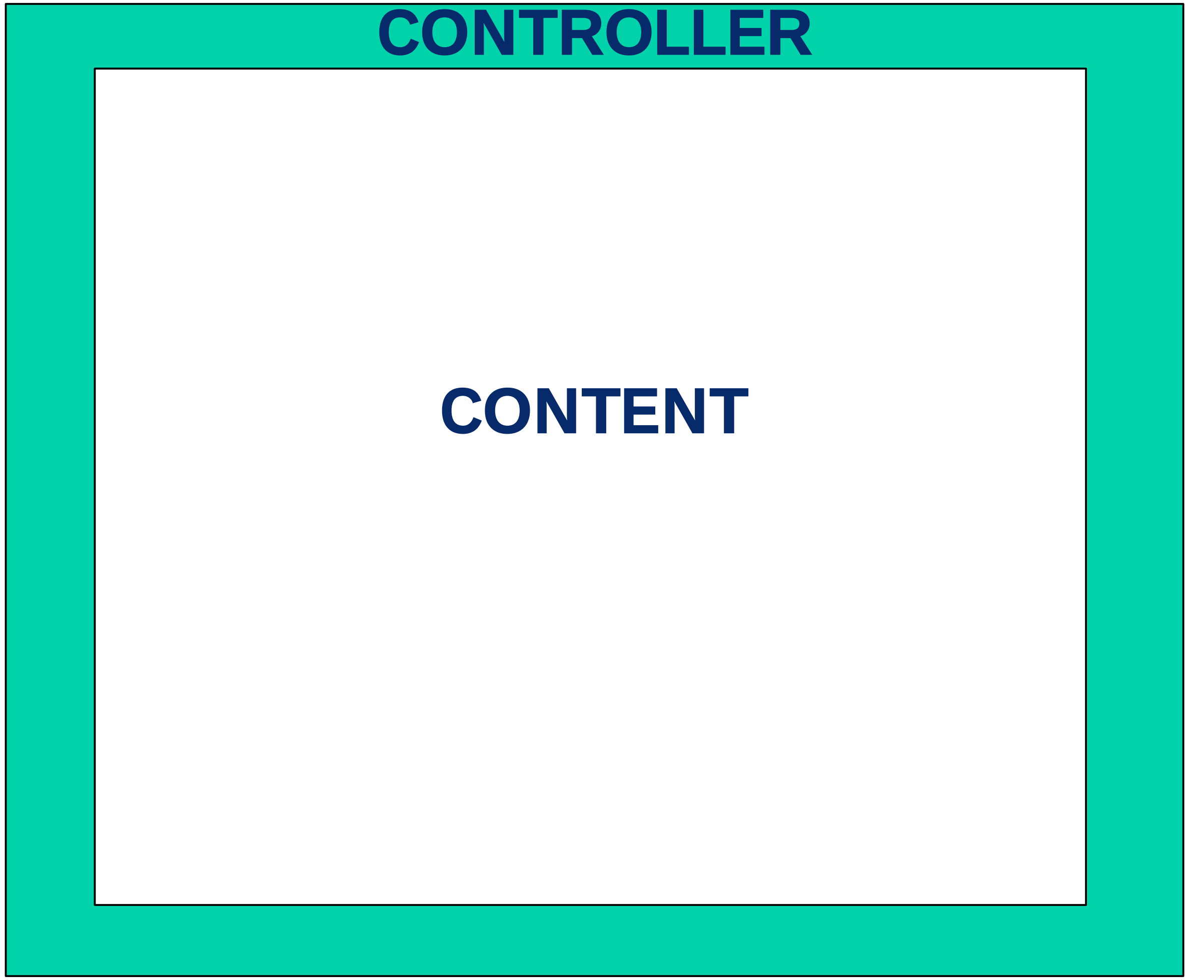
Agenda

- ❑ Component-based programming
- ❑ **Fractal component model**
- ❑ Components for Grid computing
- ❑ On-going work

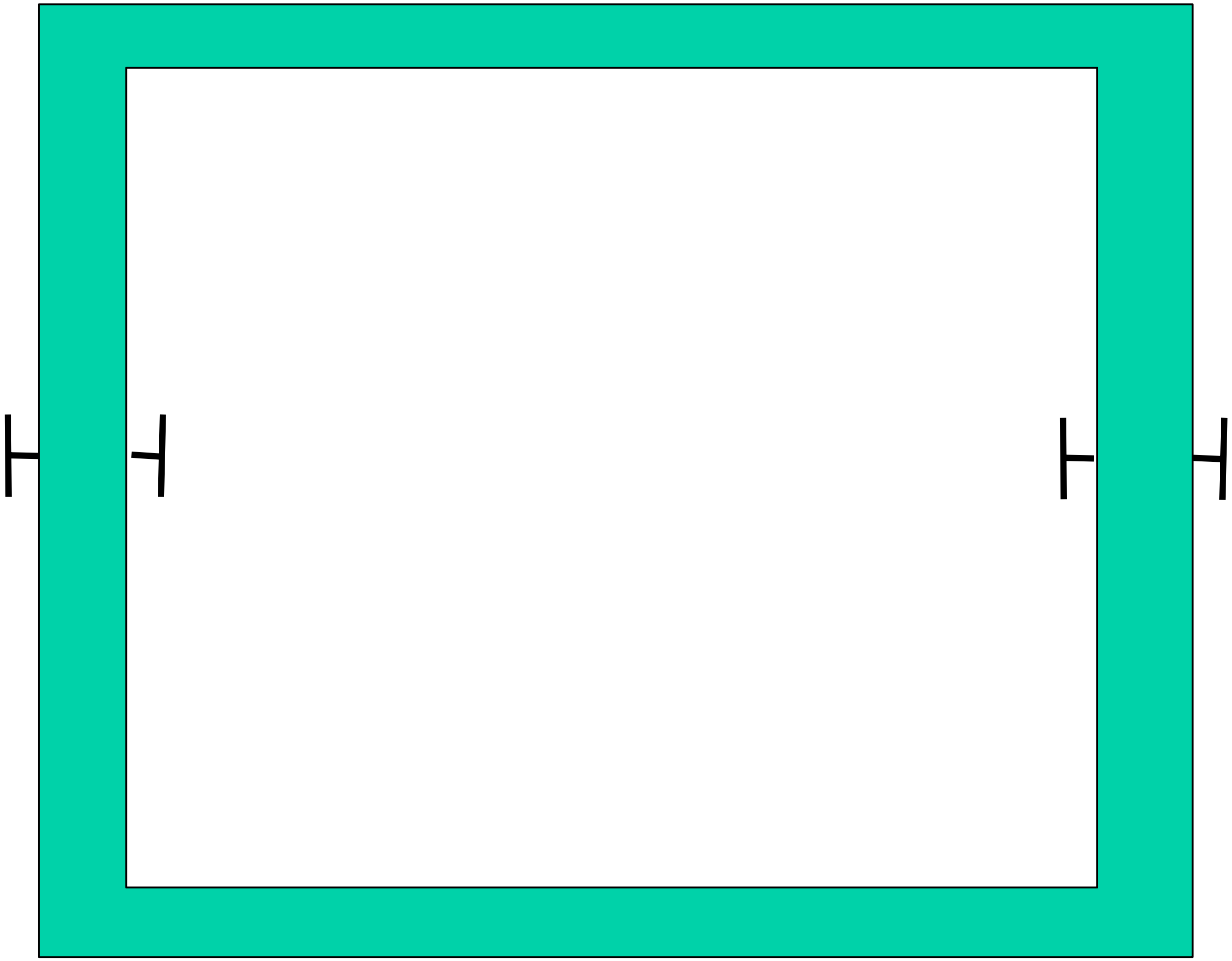
Fractal component model

- ❑ Defined by Bruneton, Coupaye, Stefani, INRIA & FT
- ❑ Key features :
  - Light
  - Extensible
  - Reflexive
  - Recursive
  - Dynamic
- ❑ Reference implementation : Julia (FT)
- ❑ New implementation based on active objects
- ❑ Standard tools in the community (ADL, GUI)

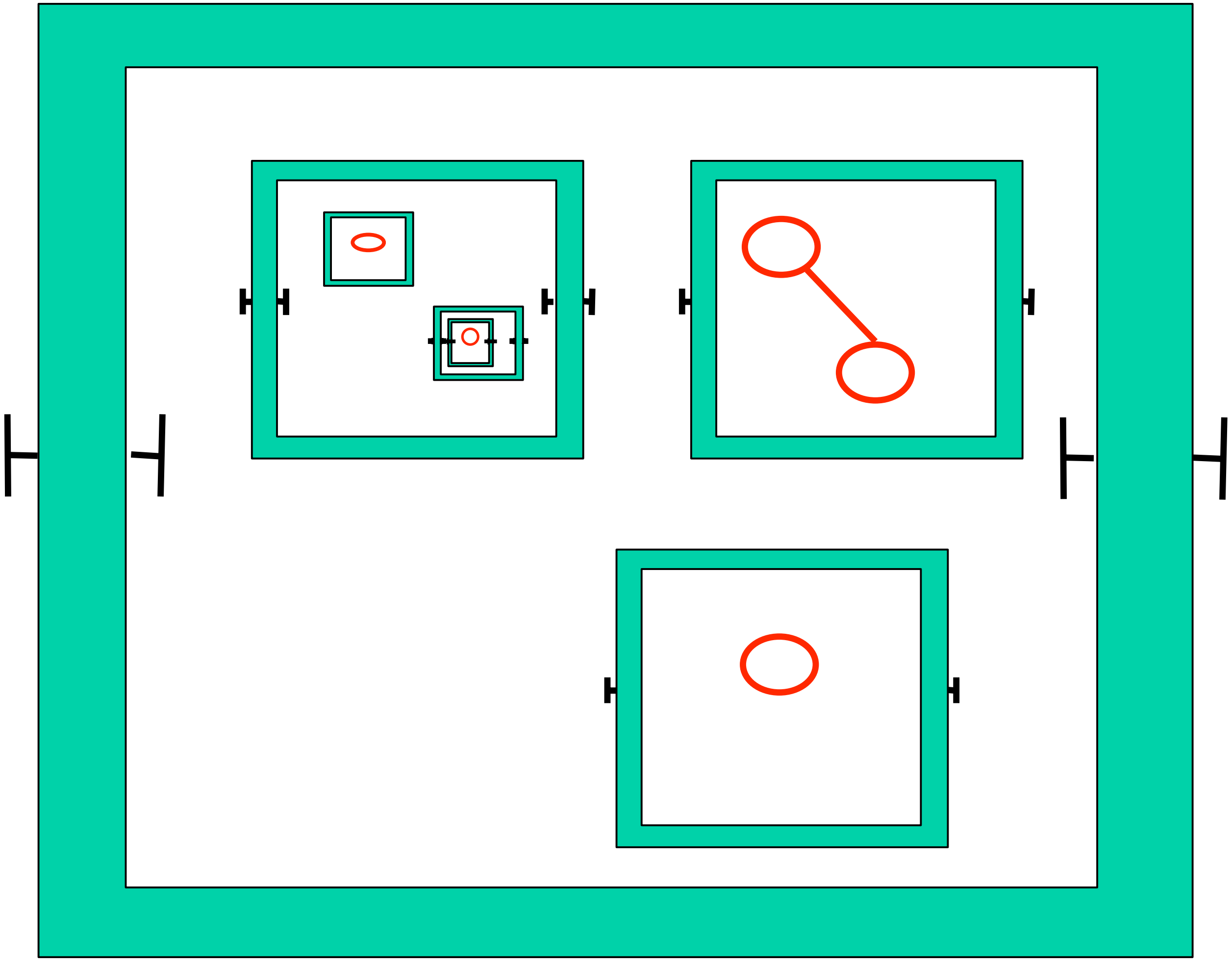
Cell analogy



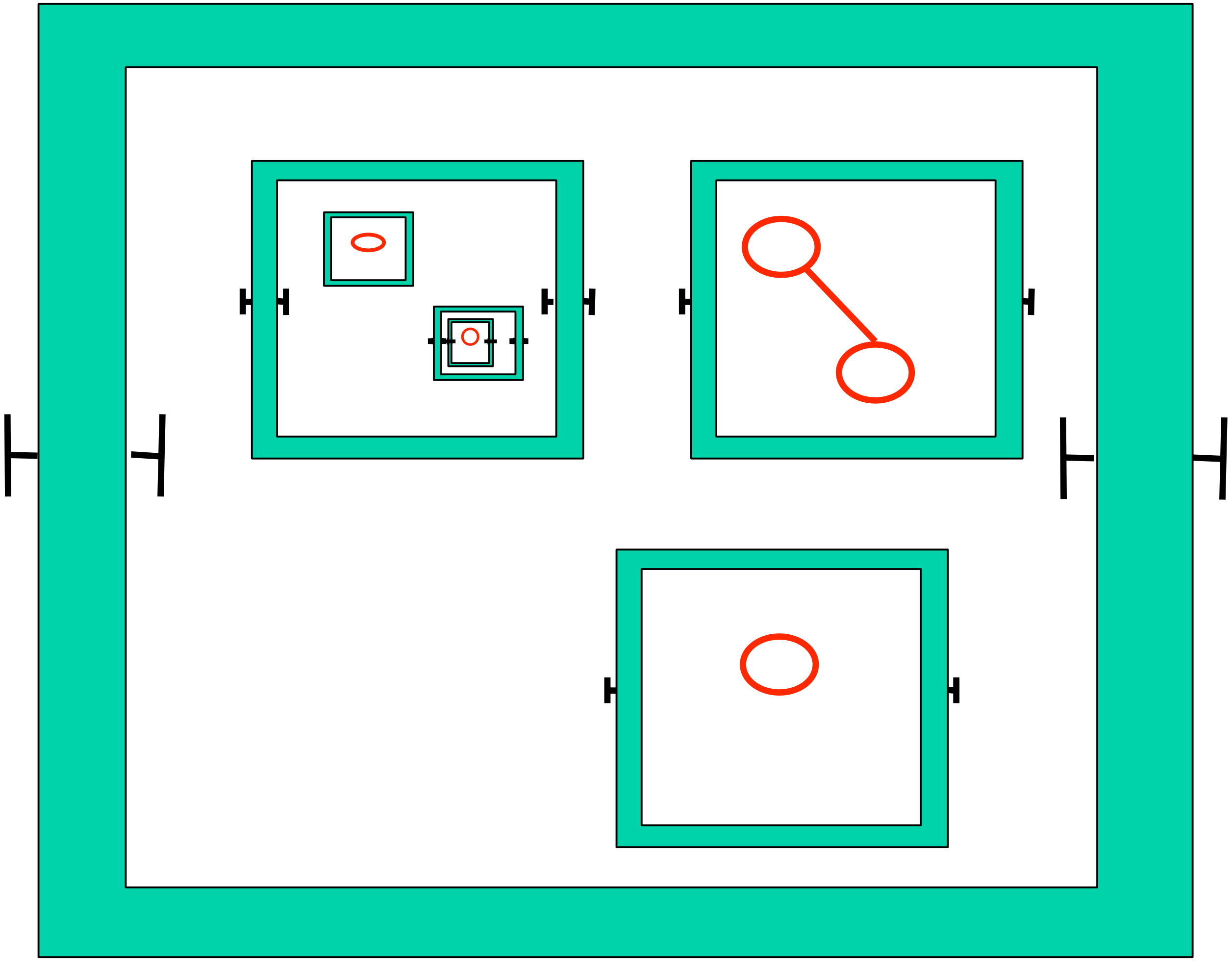
Interface = access point



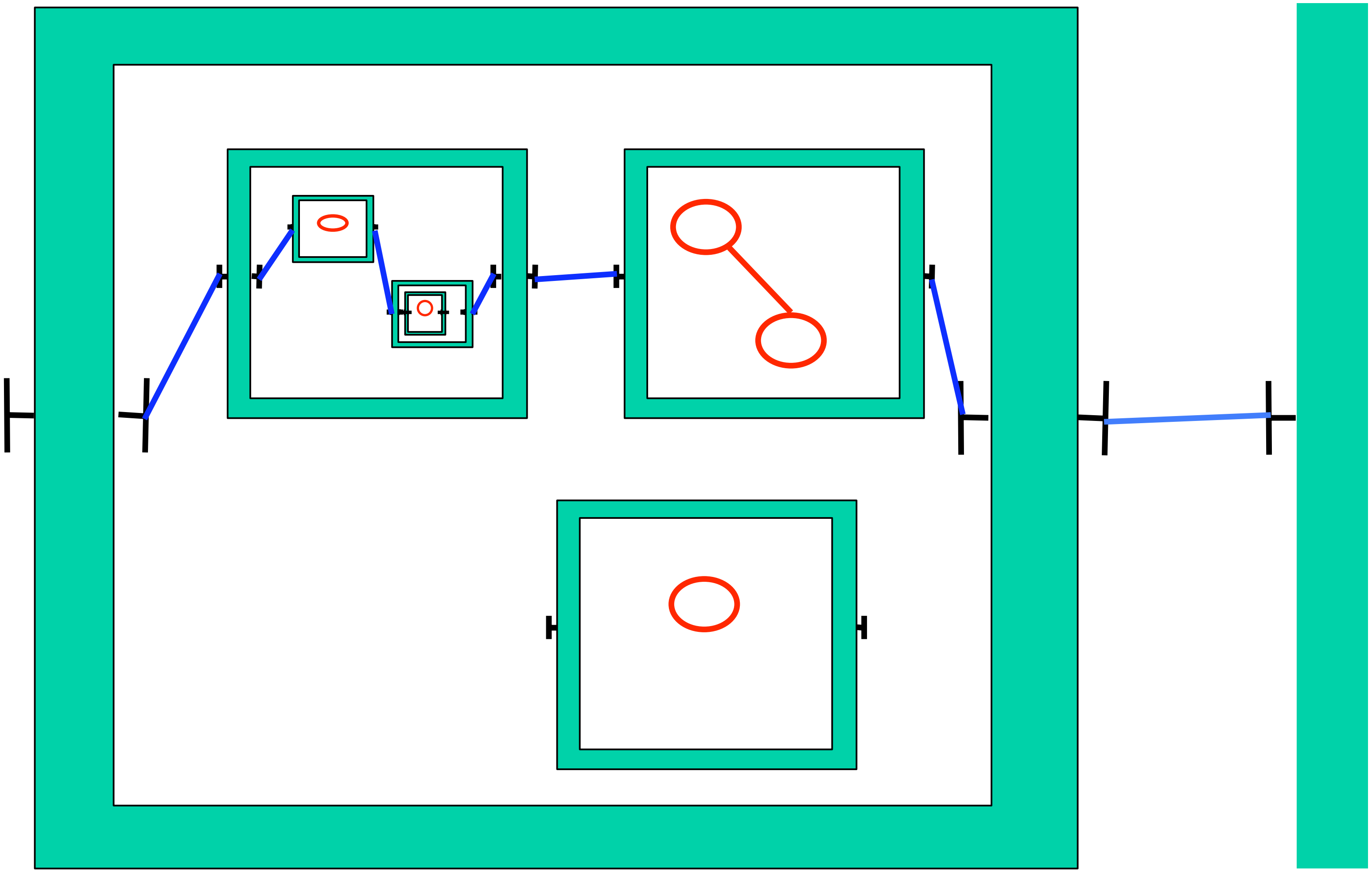
Hierarchical model



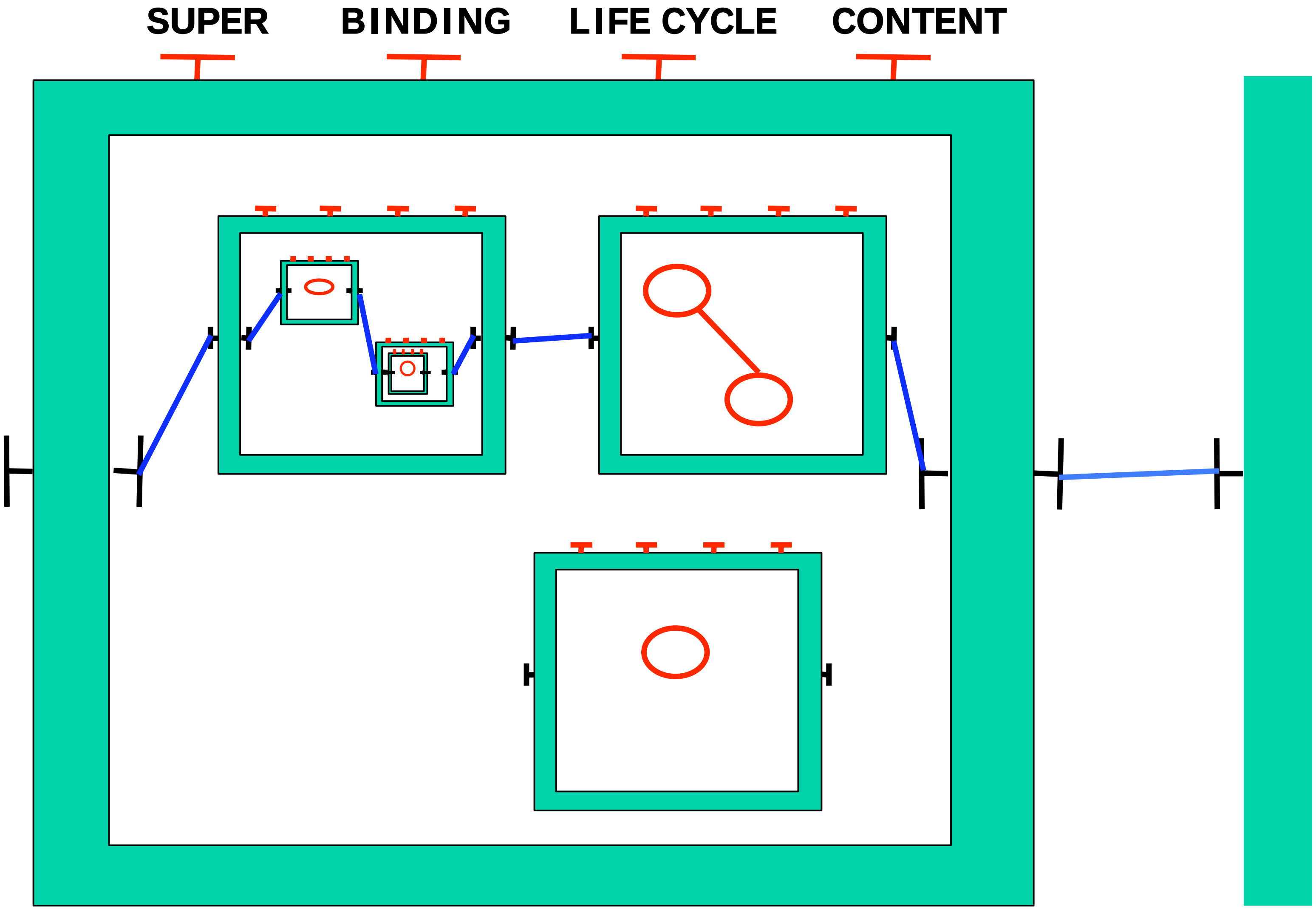
Binding = interaction



Binding = interaction



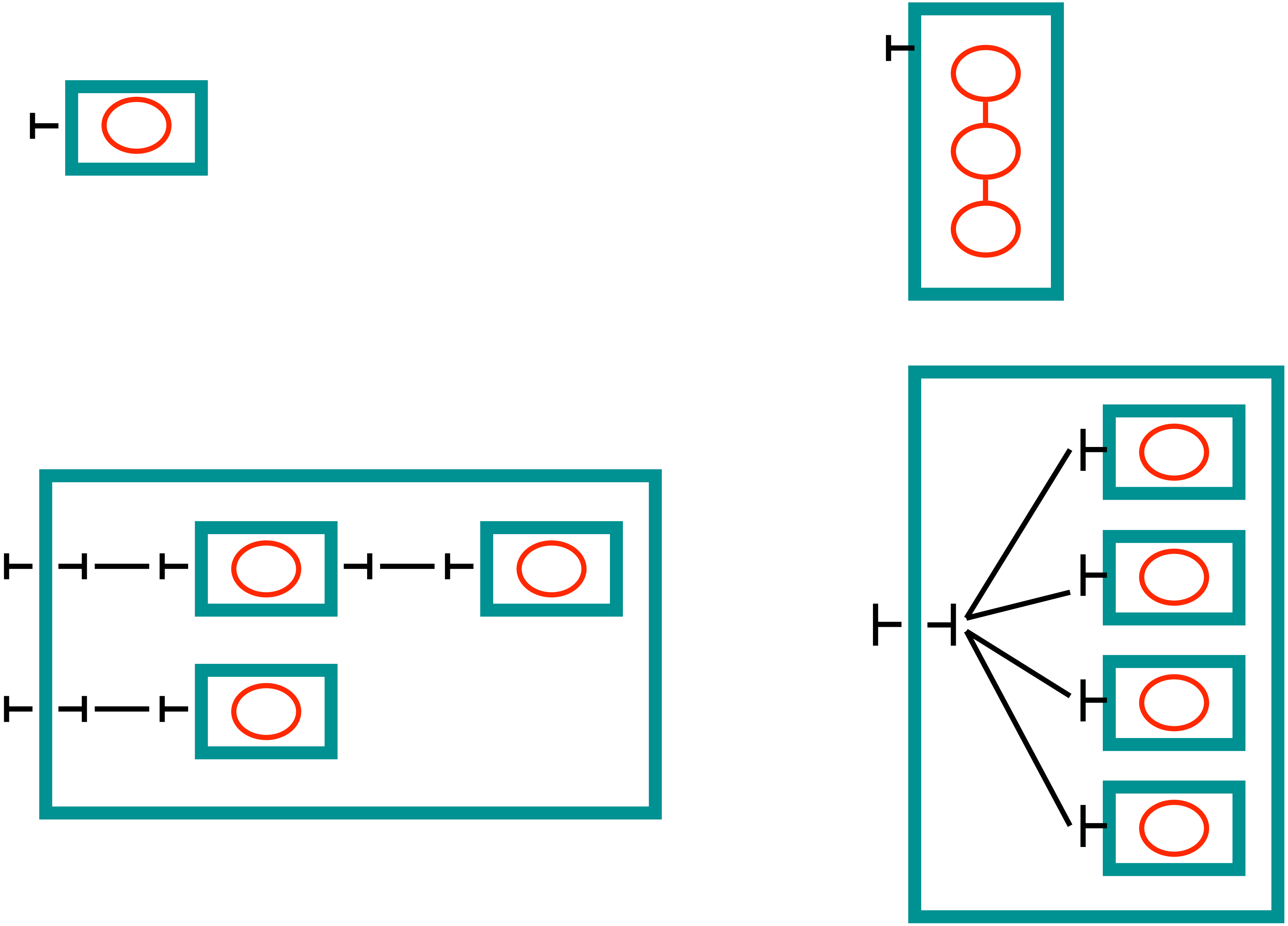
Controllers : non-functional properties



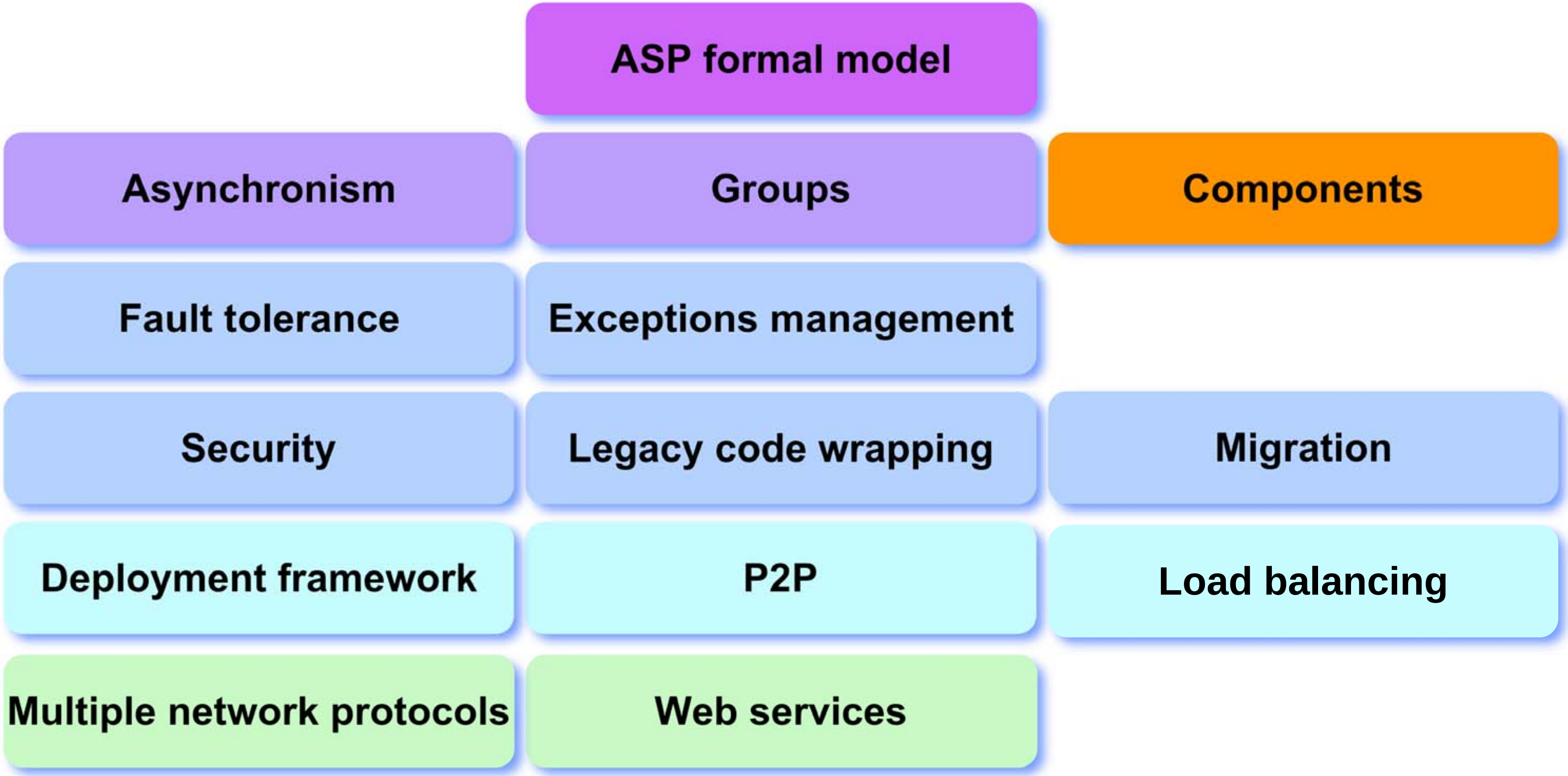
Agenda

- ❑ Component-based programming
- ❑ Fractal component model
- ❑ **Components for Grid computing**
- ❑ On-going work

ProActive components : 4 flavors



Underlying features are reusable



ASP formal model

Asynchronism

Groups

Components

Fault tolerance

Exceptions management

Security

Legacy code wrapping

Migration

Deployment framework

P2P

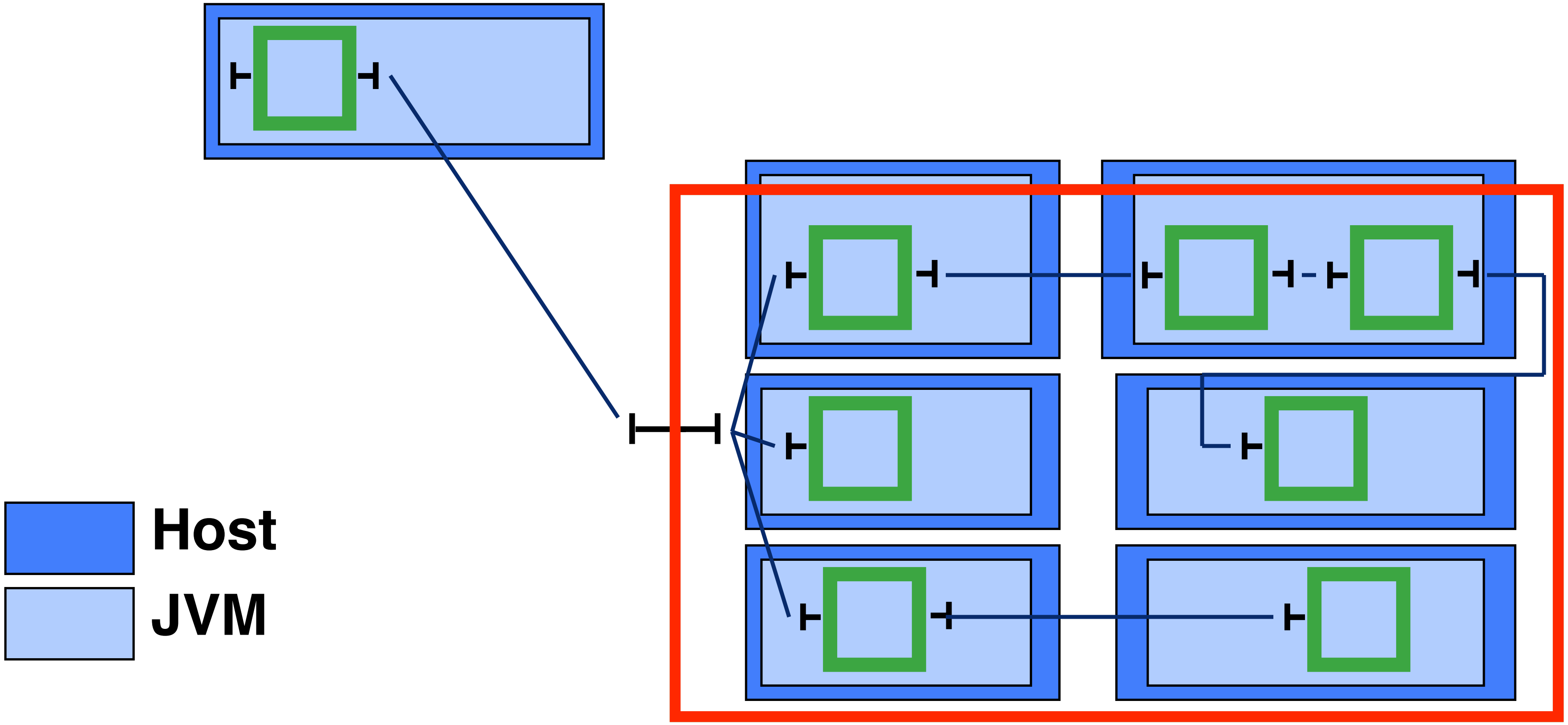
Load balancing

Multiple network protocols

Web services

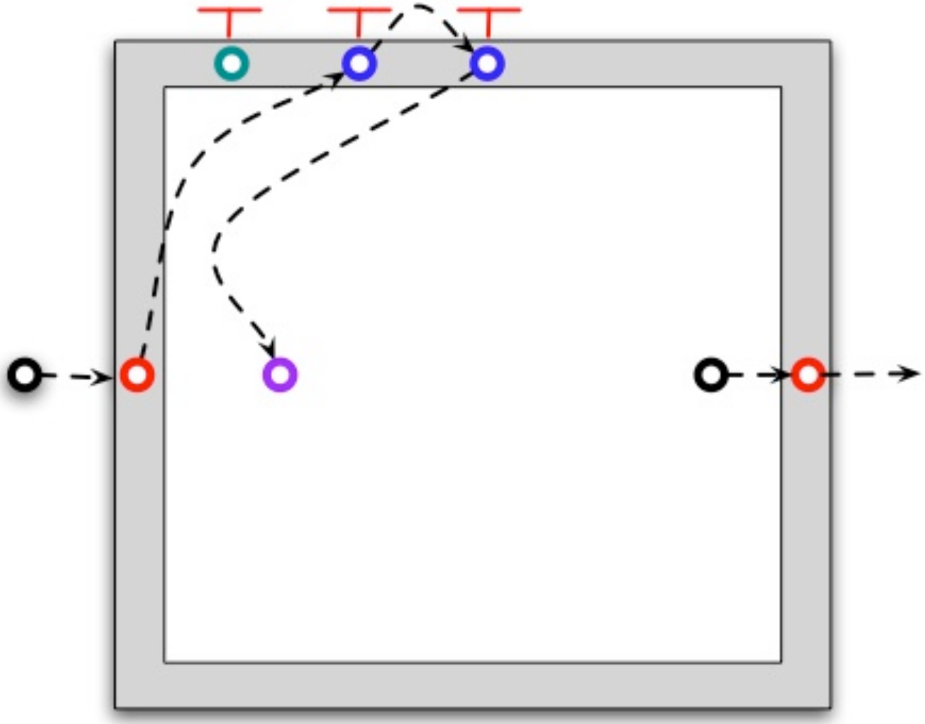
Distribution

- ❑ 1 component can be distributed over several hosts
- ❑ Distribution is transparent



Interceptions

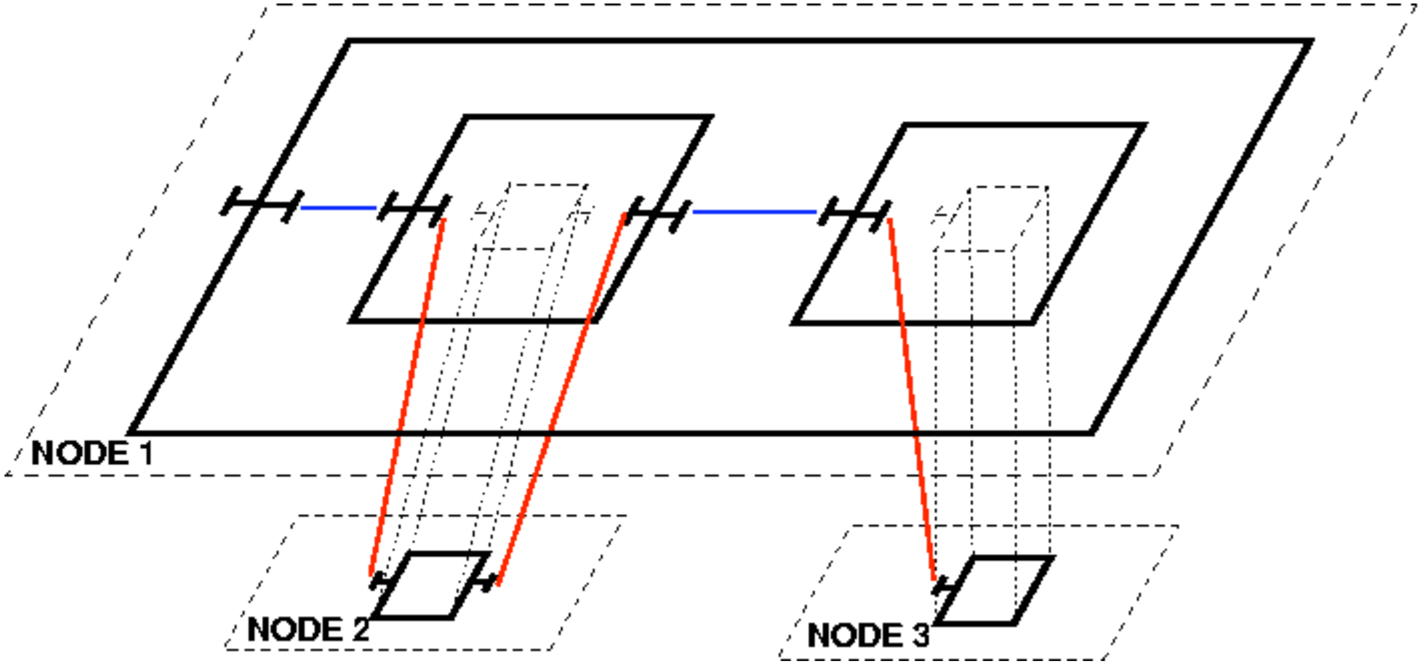
- ❑ For non-functional concerns
  - Transactions, security
- ❑ Pre / post invocations
- ❑ Input and output interceptions
- ❑ Simple composition
  - ⇨ Basic AOP



Distributed shortcuts

❑ Optimizations

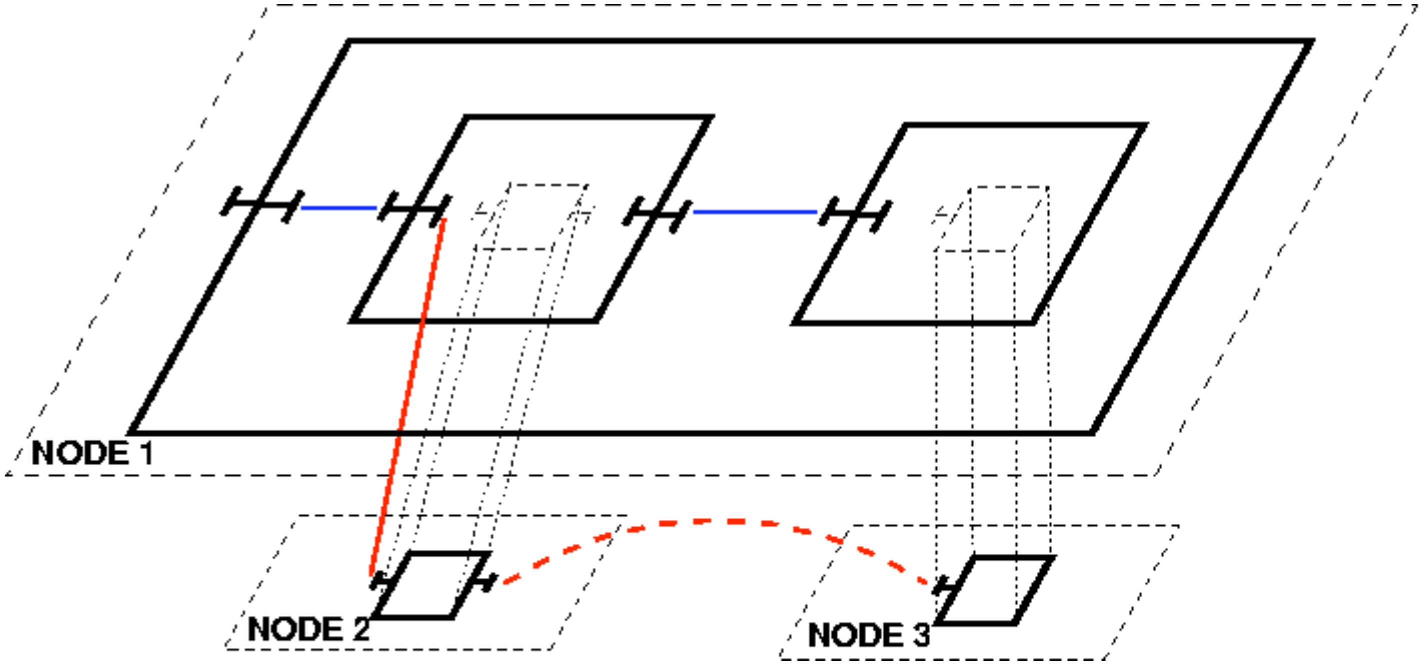
- shortcuts for distributed communications
    - distributed components



Distributed shortcuts

❑ Optimizations

- shortcuts for distributed communications
  - distributed components : **tensioning**



Web Services Integration

- Aim:
  - Expose active objects and components interfaces as Web Services

    ➔ language and technology interoperability

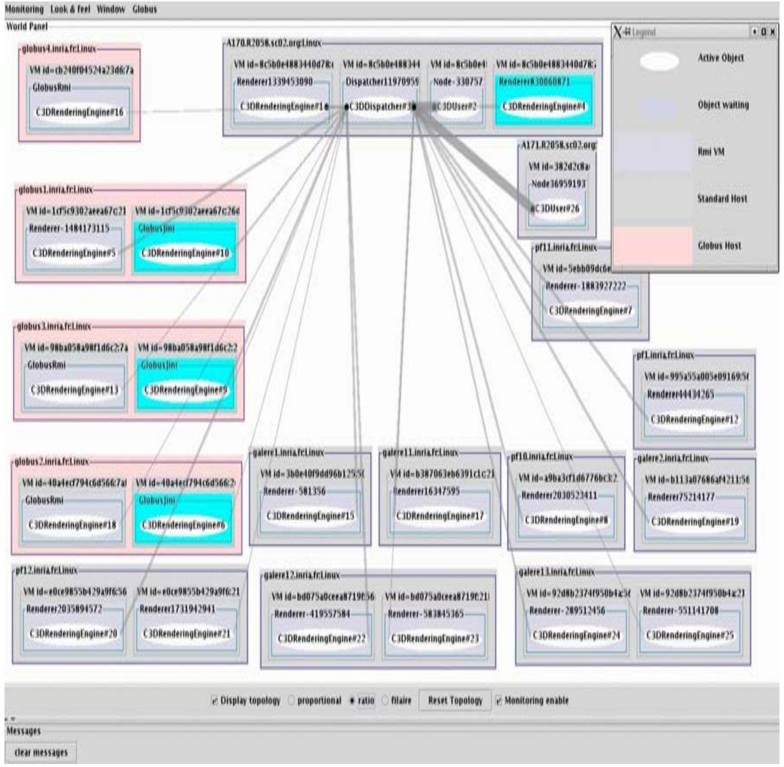
- API
  - Expose an active object as a Web Service on a web server, the user can choose the methods he wants to expose.
    - `exposeAsWebService (Object o, String url, String urn, String [] methods );`
  - Expose the interfaces of a component as Web Services
    - `exposeComponentAsWebService(Component component, String url, String componentName );`

Design and monitoring tools

- Deployment framework
  - virtual nodes
  - connection to hosts
  - creation of remote JVMs
  - instantiation / assembly / binding of components
- common ADL = common tools
  - composition of virtual nodes
  - FractalGUI
    - run-time capabilities

Design and monitoring tools

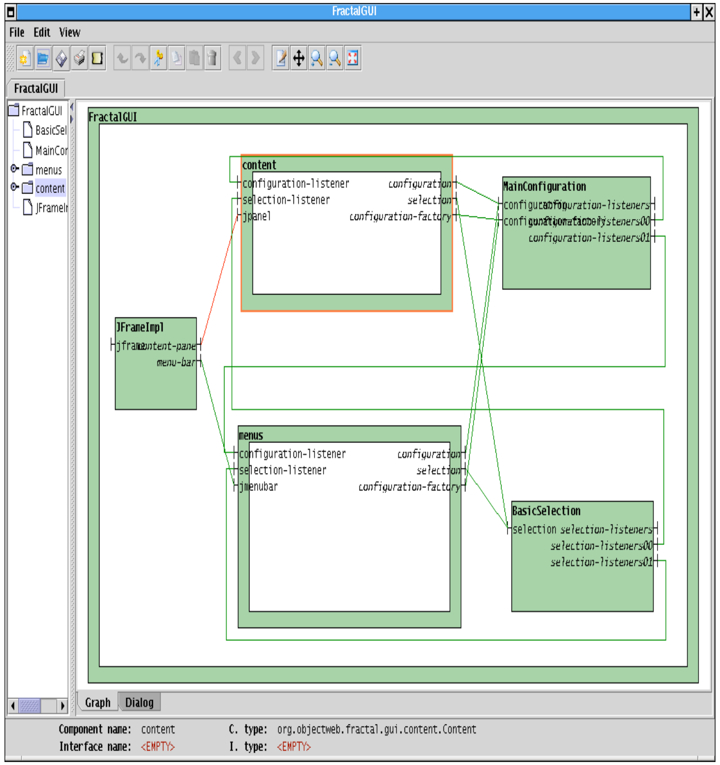
IC2D



☑ Display topology ○ proportional ● ratio ○ filaire Reset Topology ☑ Monitoring enable

U

FractalGUI



Component name: content C. type: org.objectweb.fractal.gui.content.Content
Interface name: <EMPTY> I. type: <EMPTY>

Agenda

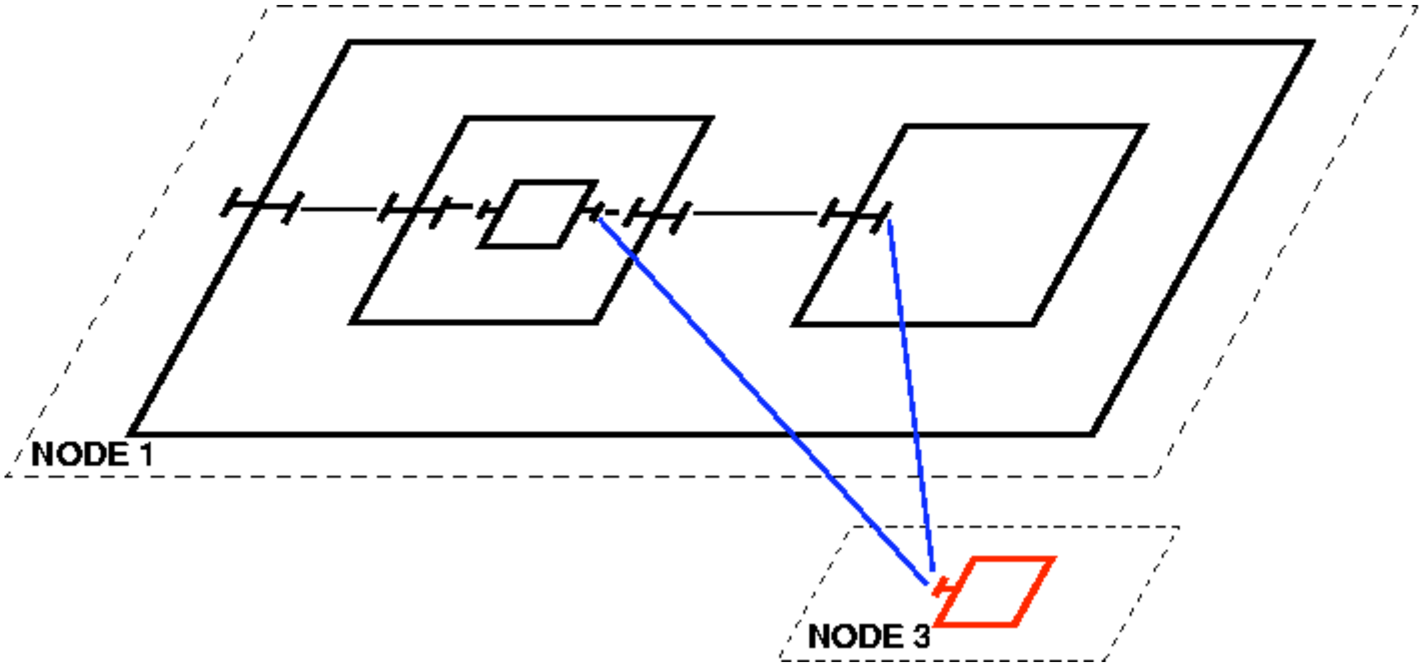
- ❑Component-based programming
- ❑Fractal component model
- ❑Components for Grid computing
- ❑**On-going work**

From objects to components

- Experiments on the refactoring of several applications
  - C3D
  - JEM3D
- Methodology
  - From objects to components
- Comparisons with object-oriented versions
  - Ease of design, modularity, extensibility

Sharing ?

- ❑A feature of the Fractal model
- ❑Currently not in our implementation
- ❑Used for representing resources (database, sensors etc...)



Dynamic reconfiguration?

- Featured in the Fractal model but :
- Asynchronous communications?
- Shortcuts ?
- Sharing ?

➔ complex operations !

➔ requires proofs through formalization

Deployment patterns

- ❑ Common topologies or assemblies automatically mapped and assembled to existing infrastructures
  - 2D grids
  - Pipelines
  - Master-slaves
  - ...

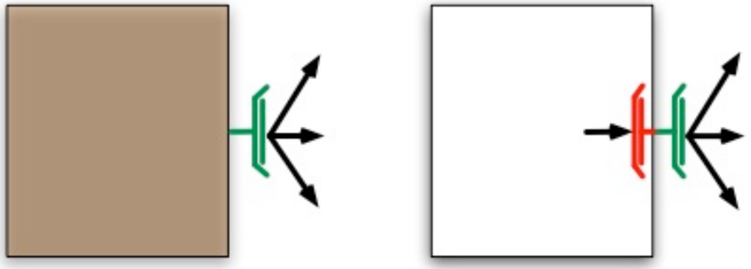
- ❑ JEM3D application as a prototype

Legacy code wrapping

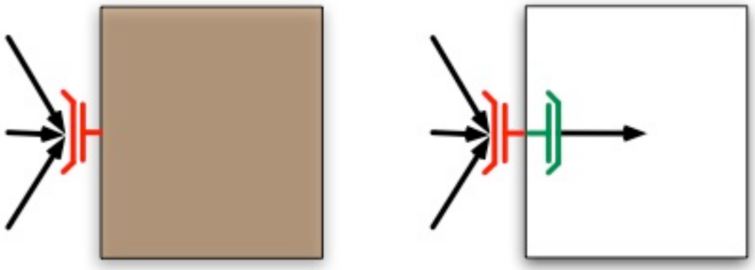
- ❑ Towards a generic mechanism
- ❑ Concepts :
  - Communication Java <--> parallel native codes --> JNI
  - Normalized interactions
    - MPI/C
    - MPI/Fortran
    - …
- ❑ First experiments for multiphysics code coupling with MPI/C (EADS)
  - Presentation by Stéphane Mariani this afternoon

Collective communications

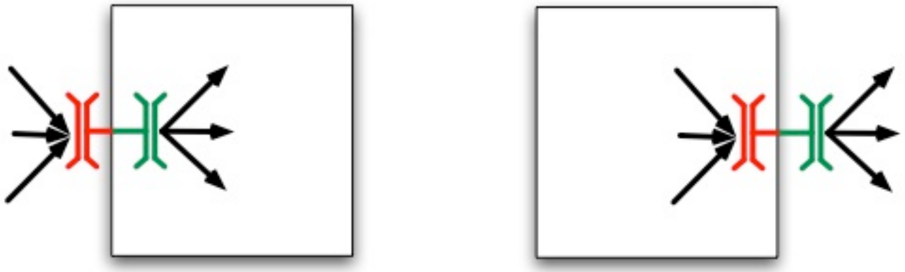
- ❑ Multicast
  - Redistribution



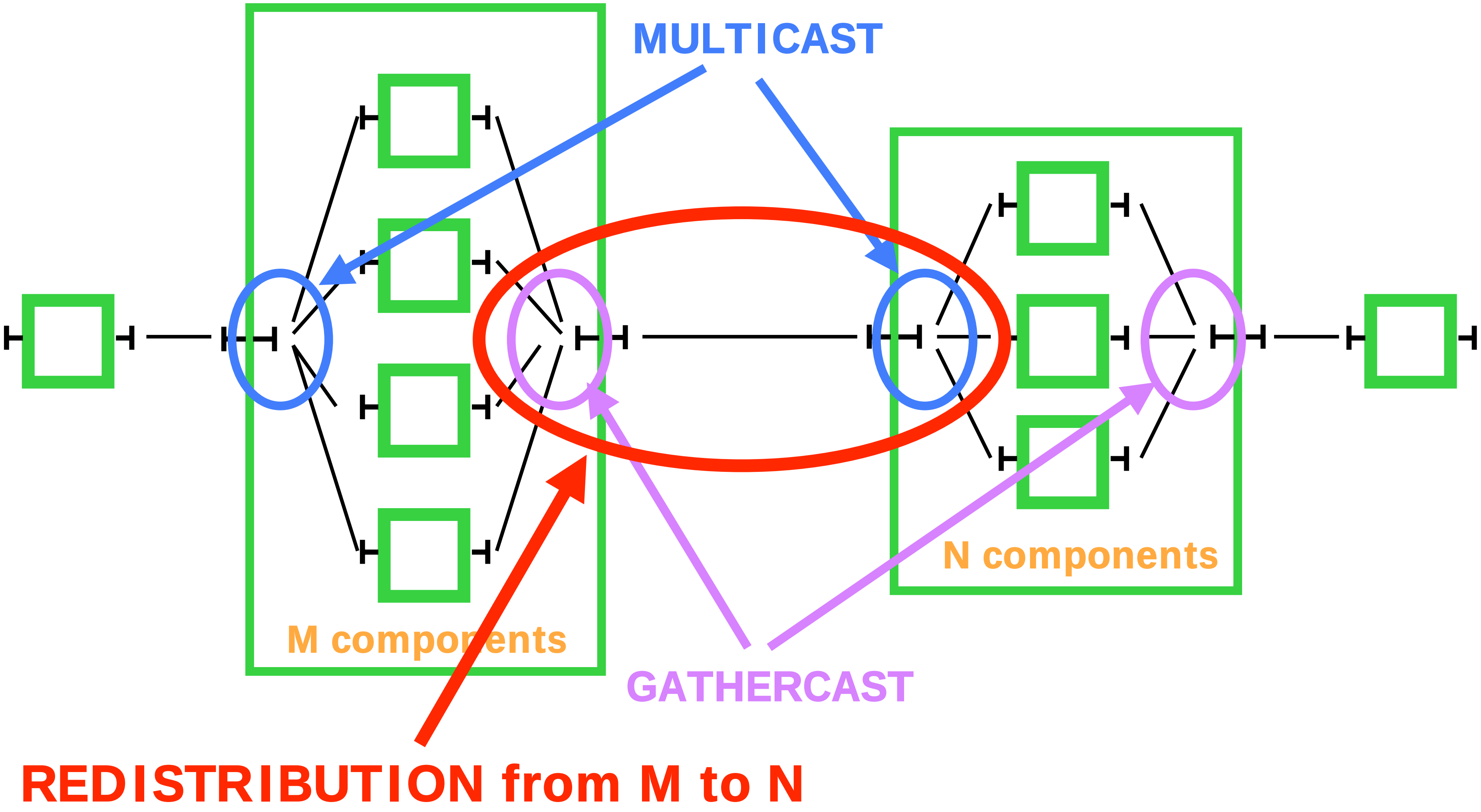
- ❑ Gathercast
  - Gathering
  - Synchronization



- ❑ Gather-multicast



MxN communications



Agenda

- ❑ Component-based programming
- ❑ Fractal component model
- ❑ Components for Grid computing
- ❑ On-going work
- ❑ **Conclusion**

Conclusion

- From a component model ...
  - Encapsulation
  - Composition
  - Description
- ... to an implementation ...
  - Based on a Grid middleware
- ... to the specification of a component model for Grid computing
  - Simple
  - Extensible (collective interfaces...)
  - Bottom-up approach

➪ easier design and management of Grid applications