An implementation of Fractal in the Think framework for building OS kernels

Aline Senart
SARDES - INRIA Project - LSR-IMAG Laboratory

Fractal Workshop
January 29th, 2002

Objective

- Development of a light-weight software architecture to allow the construction of kernels
  - that are minimal
    - As few concepts as possible
  - that are dedicated to a wide variety of hardware support
    - From PC cluster to PDA
  - that fit specific domains
    - Real-time systems, distributed systems
  - that can evolve over time
    - Replace a policy, add a functionality

Motivations

- Explosive growth in the diversity of mobile devices
  - Handheld and ubiquitous computing
- Needs to adapt to a changing environment
  - To remove network protocols in a local environment

    Modification of internal system structures

    ➪ Needs to exhibit the architecture
  - To increase security when reaching an untrusted network

    Change a policy dynamically

Outline of the talk

- Think component-based architecture
  - An overview of the software framework
  - Implementation of the framework
- Fractal in the Think framework
  - Use of Fractal for building OS kernels
  - Experiments in using Fractal
  - An implementation of Fractal
- First evaluation

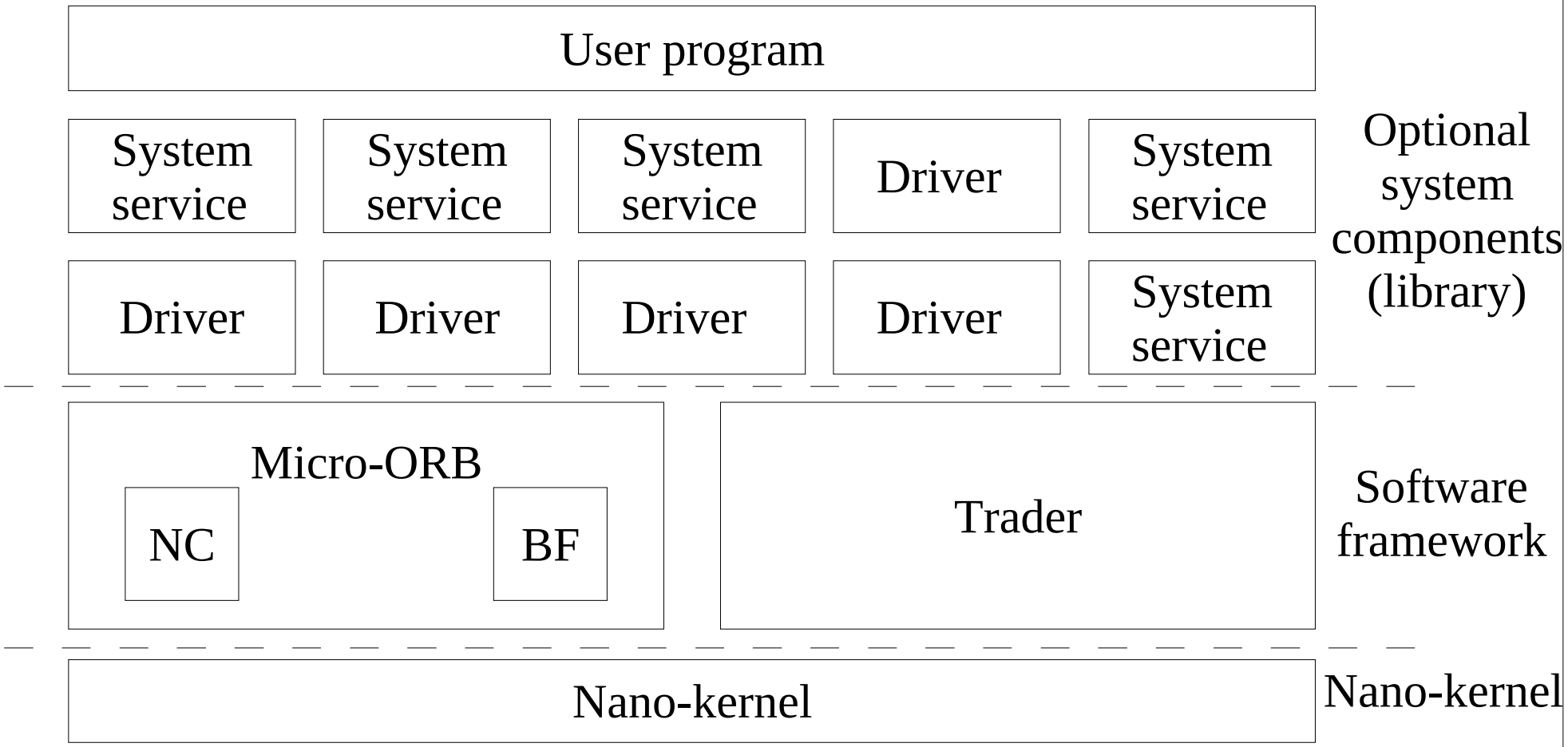
Think Is Not a Kernel

An overview of the software framework

- ☞ *A component-based architecture dedicated to build customized OS kernels*
  - A minimal nano-kernel highly dependent of hardware support
    - Hardware resources reified (ex : processor)
  - A set of optional system components in a library
    - Device drivers and service systems (ex : scheduler)
  - A software framework
    - Definition of a programming model inspired by the standard ODP reference model
      - Basic modeling concepts : components, interfaces, bindings, naming contexts, binding factories

How to build a kernel with Think

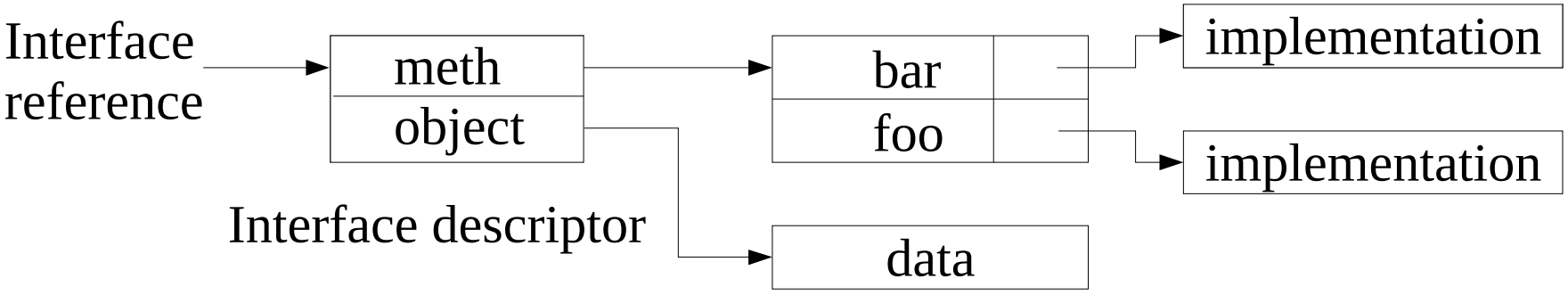
An overview of the software framework



Instantiation of interfaces

Implementation of the framework

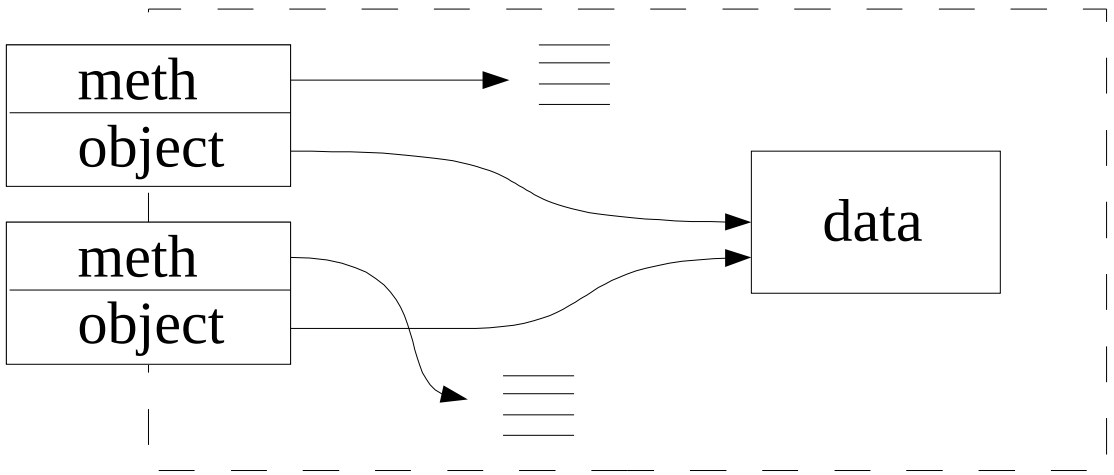
- Component is a run-time structure
  - Interface = Named interaction point
  - Binding = Interaction object between interfaces
- Run-time representation of an interface



Instantiation of components

Implementation of the framework

- A component owns one or more functional interfaces
  - Pointer to interface methods implementation
  - Reference to a set of instance variables
    - Data shared amongst all interfaces



Outline of the talk

- Think component-based architecture
  - An overview of the software framework
  - Implementation of the framework
- Fractal in the Think framework
  - Use of Fractal for building OS kernels
  - Experiments in using Fractal
  - An implementation of Fractal
- First evaluation

Why Fractal?

Use of Fractal for building OS kernels

- Standardization of component models within ObjectWeb consortium
- A software framework
  - Minimalist approach
- Component-based infrastructure
  - Configuration
    - Hierarchic composition model
  - Reconfiguration
    - Addition/suppression of components, interfaces...

Our contribution

Use of Fractal for building OS kernels

➪ Enhancement of the Think framework with a C implementation of a mini-Fractal

- Fractal composition model
  - Recursive with a content/controller pattern
- Fractal core framework for reconfiguration
  - ☞ Meta-level interfaces to control components' behavior
  - Identity, content, binding, lifecycle

Main difficulties

Experiments in using Fractal

- OS domain requires a lightweight support
  - Main Fractal programming interfaces
  - Paying reconfiguration cost only if a reconfiguration takes place
- Without predefined assumptions
  - Each component must be free to define its own control policies
- Keeping the existing system
  - Allow the coexistence of both component models

Comparaison with Fractal specifications

Experiments in using Fractal

- Names in BindingController and ComponentIdentity

```
void bind(Name client, Name server);
void unbind(Name client);
```

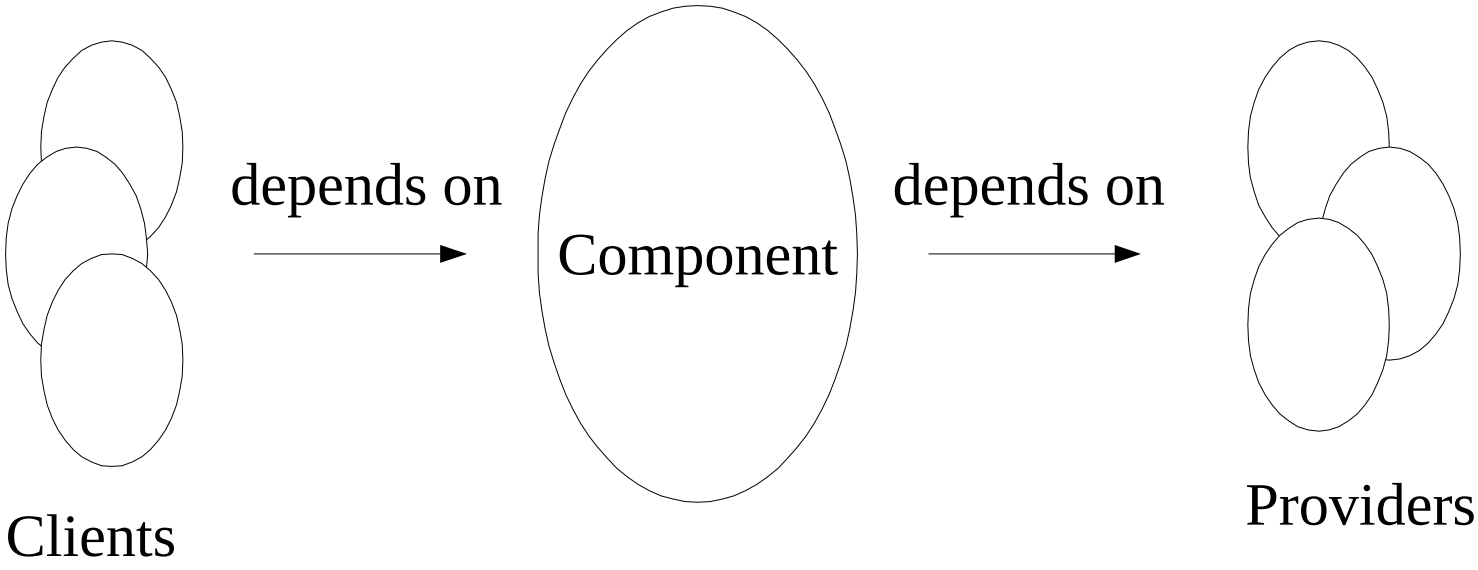
```
Name getInterfaces(String name);
Name[] getInterfaces();
```

InterfaceReference not present in Think

Comparaison with Fractal specifications

Experiments in using Fractal

- Names in BindingController and ComponentIdentity
- New controller interface : ClientController



To ease the deletion of server components

Comparaison with Fractal specifications

Experiments in using Fractal

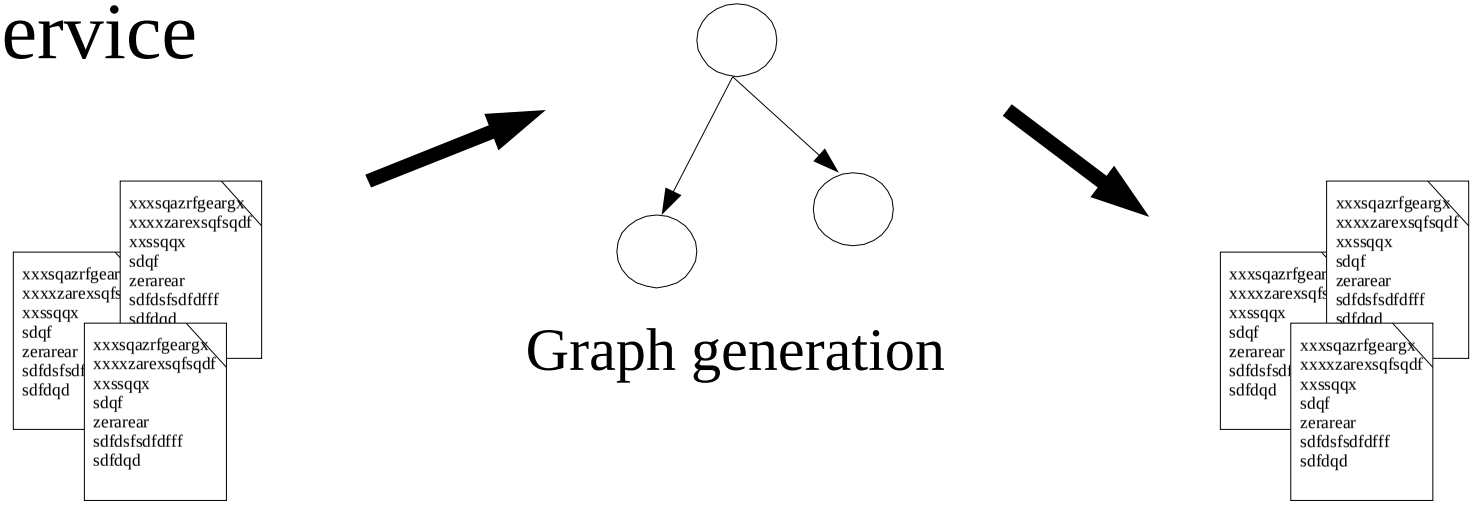
- Names in BindingController and ComponentIdentity
- New controller interface : ClientController

To ease the deletion of server components

Comparaison with Fractal specifications

Experiments in using Fractal

- Names in BindingController and ComponentIdentity
- New controller interface : ClientController
- Deployment process : automatic configuration service



A description for each component

Generation of program skeletons with a default implementation

Comparaison with Fractal specifications

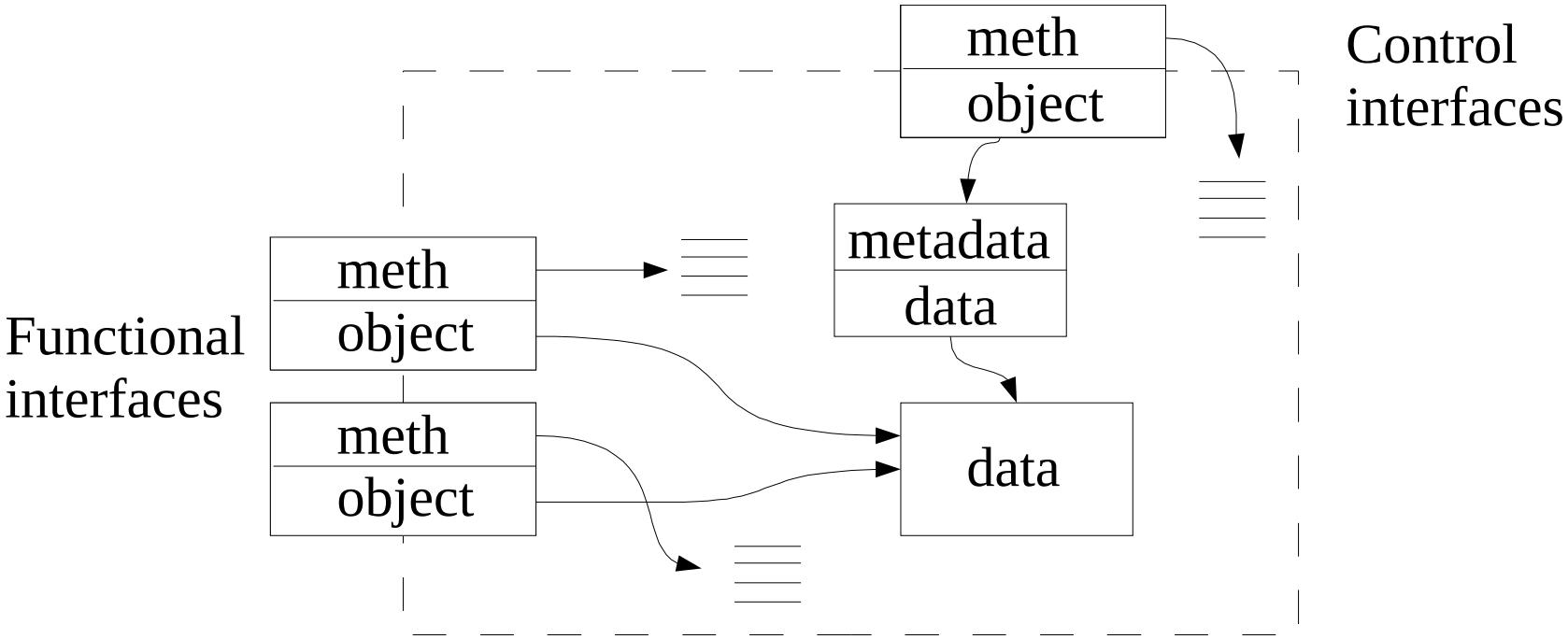
Experiments in using Fractal

- Names in BindingController and ComponentIdentity
- New controller interface : ClientController
- Deployment process : automatic configuration service
- No exception raised and caught

A reconfigurable component

An implementation of Fractal

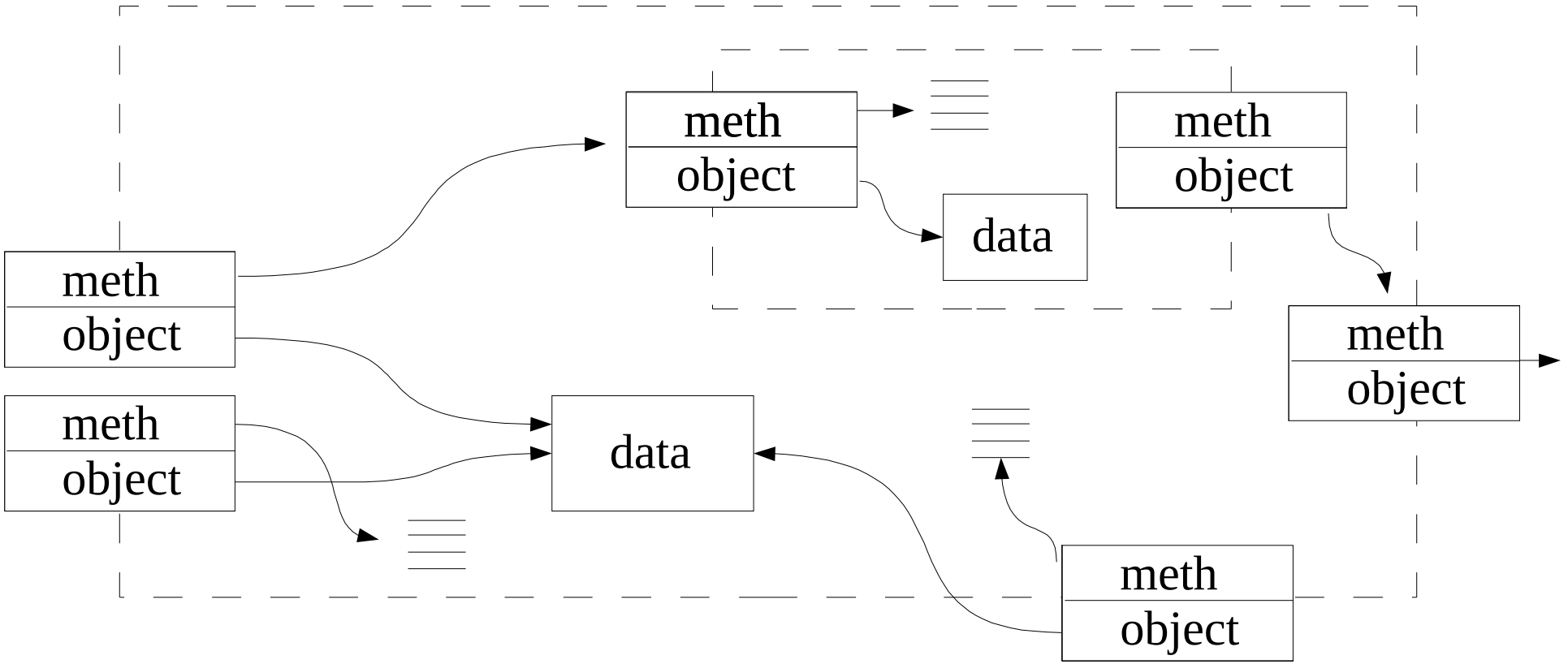
- A reconfigurable component can define some control interfaces
  - This requires to store meta-data on the system



A composite component

An implementation of Fractal

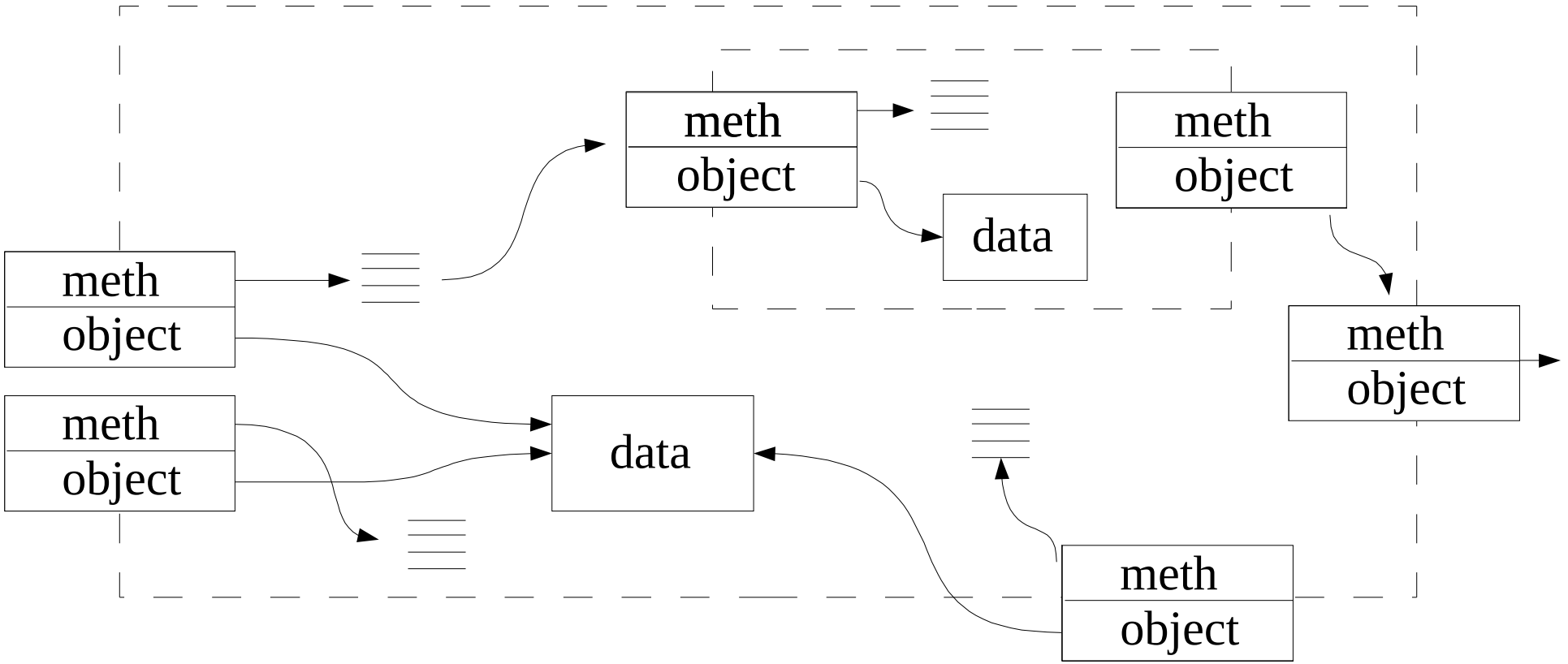
- Visibility of interfaces determined by the enclosing component



A composite component

An implementation of Fractal

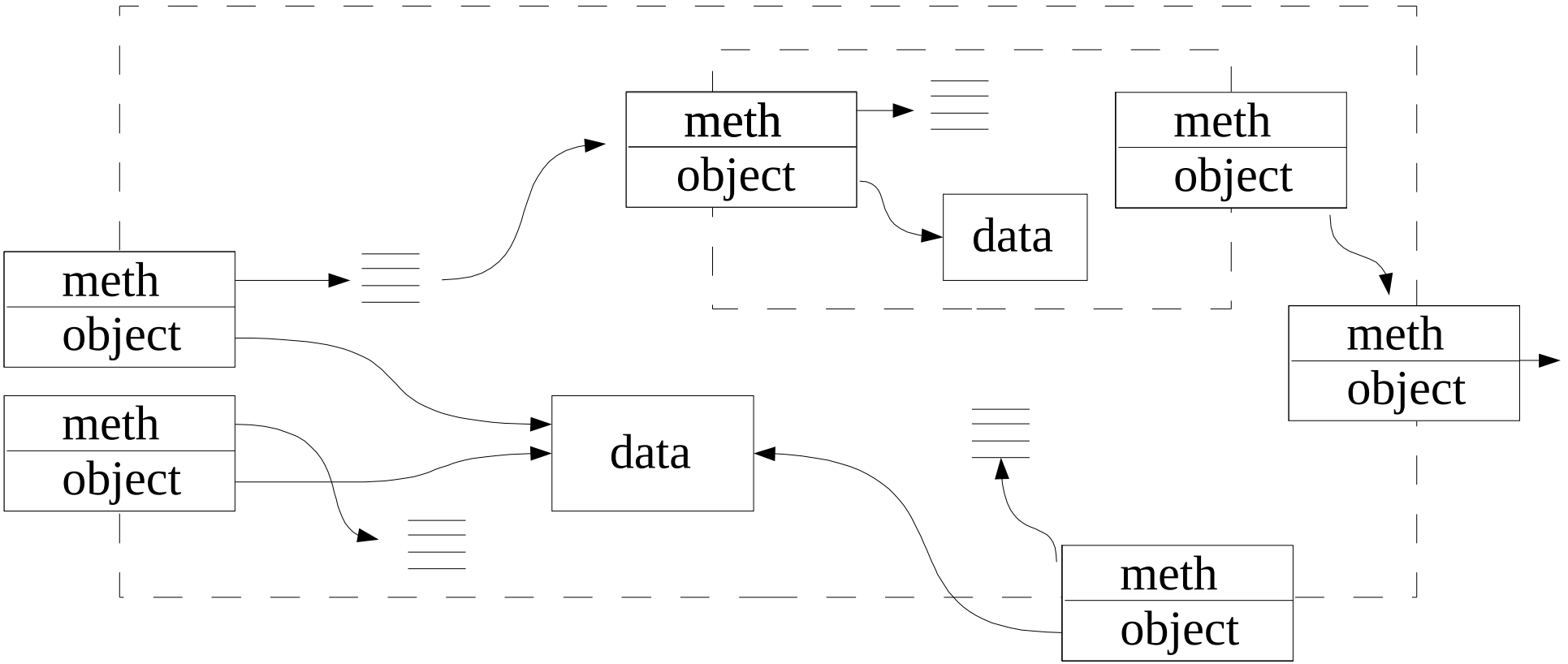
- Visibility of interfaces determined by the enclosing component



A composite component

An implementation of Fractal

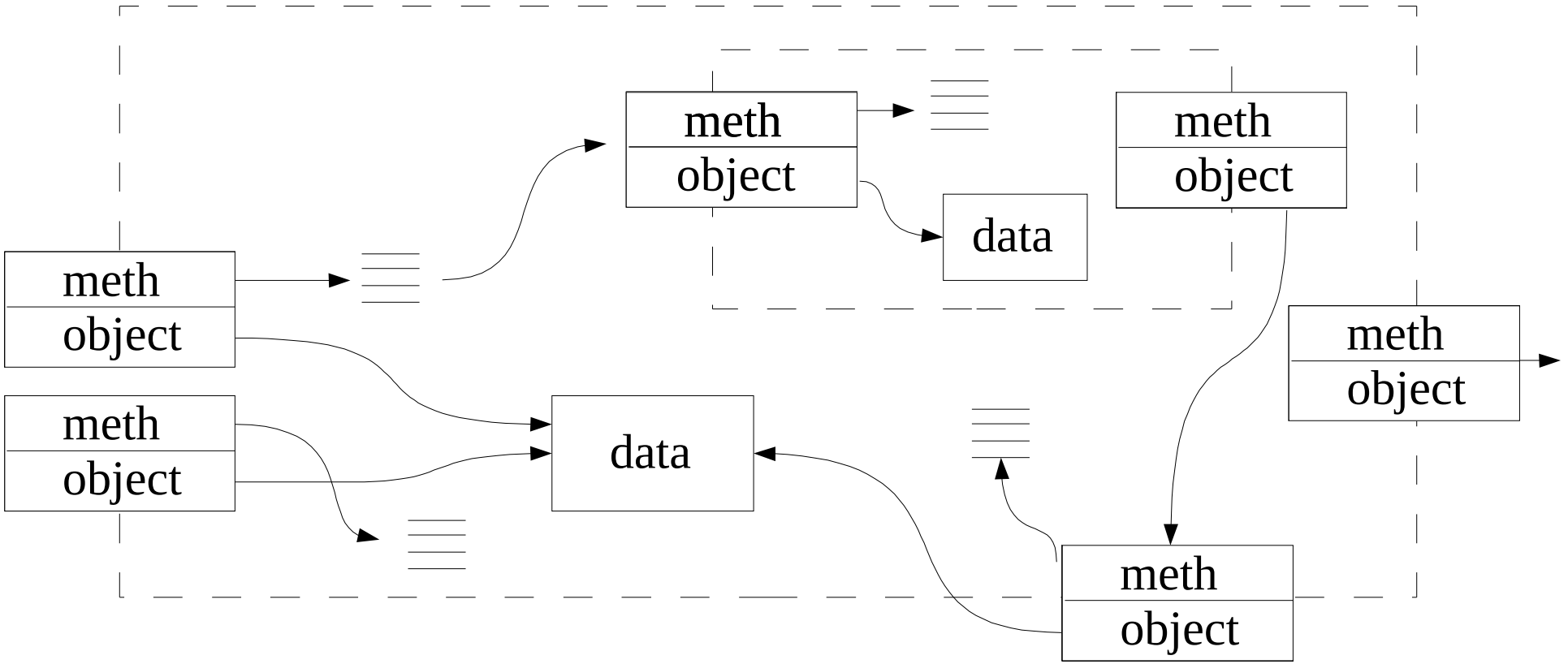
- Visibility of interfaces determined by the enclosing component



A composite component

An implementation of Fractal

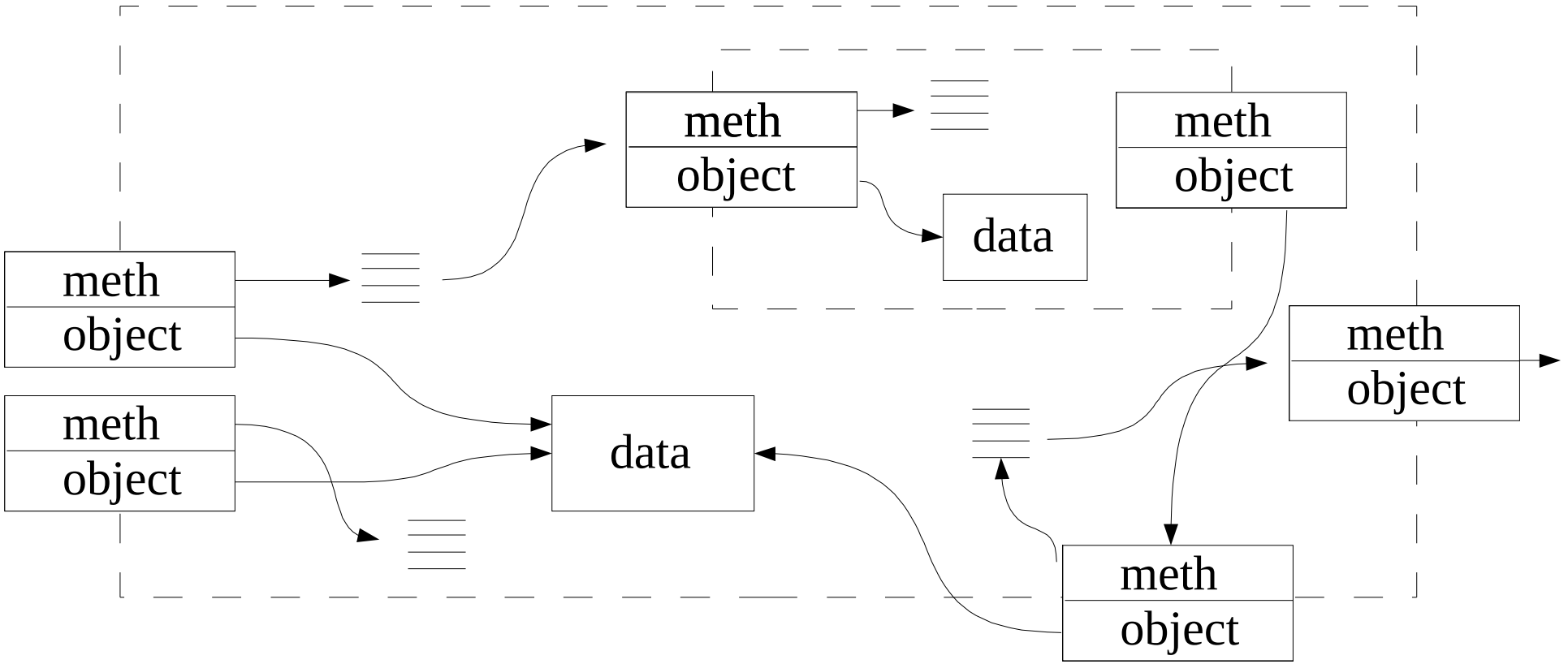
- Visibility of interfaces determined by the enclosing component



A composite component

An implementation of Fractal

- Visibility of interfaces determined by the enclosing component



Outline of the talk

- Think component-based architecture
  - An overview of the software framework
  - Implementation of the framework
- Fractal in the Think framework
  - Use of Fractal for building OS kernels
  - Experiments in using Fractal
  - An implementation of Fractal

➢ First evaluation

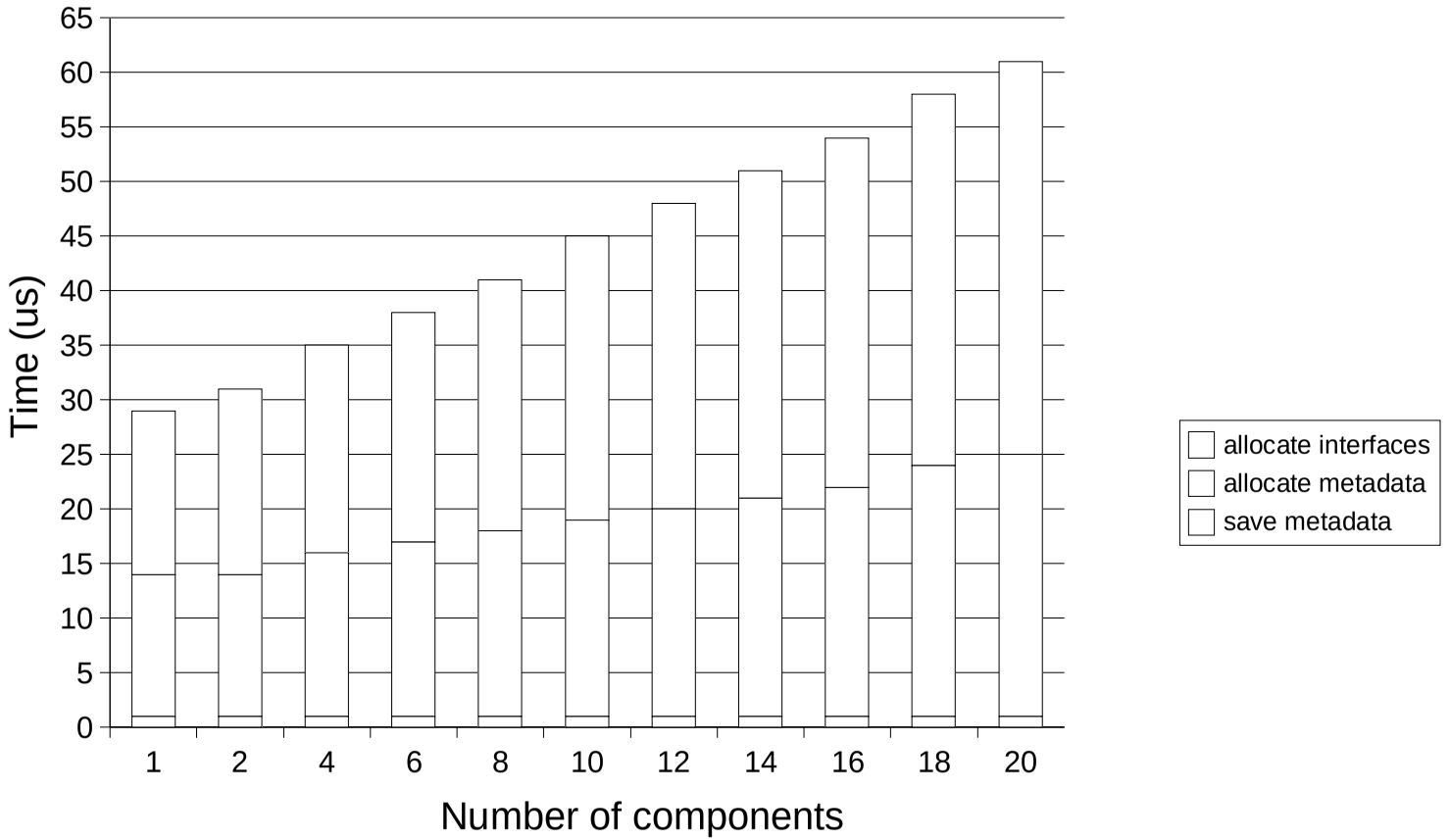
Experimentations

Preliminary experiments on a PowerPC G3 running at 400 Mhz with 128 MB memory

- Slight increase of memory (less than 5 %)
- No cost added in the global execution time when no dynamic changes are performed (factorial, quicksort algorithms)
- Framework greatly eases the development of reconfigurable kernels

Instantiating components

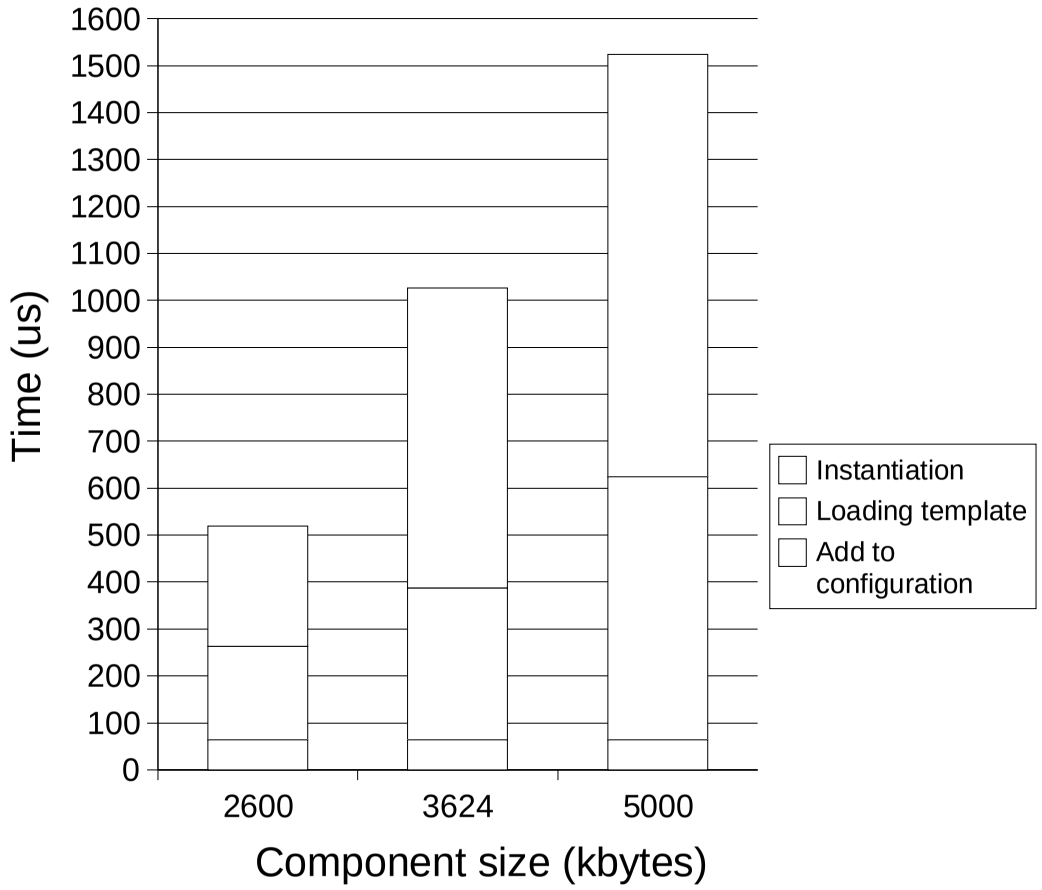
Performances



Multiple instantiations of a 3KB component

Loading components of different sizes

Performances



Loading template

- Mounting file system via e2fs
- Reading and loading the template in memory

Instantiation

- Dynamic memory allocations
- Metadata storage

Add to configuration

- Composite metadata updated

Conclusion

- Selection of a component-based framework
  - THink Is Not a Kernel
- Enhancement of the component model
  - Uniform approach: all entities in the system are components
- Conception of a reconfiguration framework
  - Dynamic configuration
  - Full run-time introspection
- Perspective: to experiment the framework in active networks domain