The Fractal Open Component Model

Jean-Bernard Stefani

INRIA Grenoble-Rhône-Alpes
Executive Summary

- **Fractal**: a model for the construction of reconfigurable component-based software systems
- **Fractal**: an open model
  - different forms of composition
  - different forms of reflection
  - different forms of connection
  - programming-language independent
- **Fractal**: an open source project, hosted by the OW2 consortium
  - Model specifications
  - Design, programming & deployment tools
  - Industrial-strength Fractal-based projects
1. CBSE and Fractal

2. The Fractal Component Model

3. The Fractal Project

4. Fractal-based Developments

5. Conclusion
Outline

1. CBSE and Fractal
2. The Fractal Component Model
3. The Fractal Project
4. Fractal-based Developments
5. Conclusion
CBSE: branch of software engineering that studies the design and construction of software systems as explicit compositions of software units (components).

- Semantical foundations: components and composition operators
- Design and programming tools: architecture description and analysis
- Pattern catalogs: architecture styles

Components: units of (data and behavior) encapsulation with well defined interfaces for communication with other components.

- In contrast to objects, components expose as interfaces all their communication relations.
Motivations for Fractal

- Component-based software engineering (CBSE) for software system construction and management
  - Components as units of modularity, fault isolation, distribution, configuration, deployment
  - Software architecture central for design, construction and management

- Question #1: how to support CBSE and its multiple concerns?
Motivations for Fractal

“Components” among us
  - plugins, xBeans, packages, COM & .Net, OSGI, SCA, etc

At the crossroad of multiple, overlapping concerns
  - modularity
  - architecture
  - software evolution
  - distribution
  - deployment
  - dependability

Question #2: how to understand the diversity of components?
Motivations for Fractal Answering questions #1 and #2 by combining in the same component model:

- Sophisticated architecture description capabilities
  - component hierarchies
  - sharing graphs
- Multiple forms of component composition and component interaction
  - concurrency and control
  - communication and synchronization
- Multiple forms of reflection
  - interception
  - introspection
  - superposition
Outline

1. CBSE and Fractal
2. The Fractal Component Model
3. The Fractal Project
4. Fractal-based Developments
5. Conclusion
Component

- Encapsulated data and behavior
- With well identified interfaces
- With sub-components

Interface

- A named access point to a component
- Can emit or receive operations or messages
  - E.g. client and server interfaces
- Can be typed

Binding

- Communication path between components
- Bindings mediate all interactions between components
Component = membrane + sub-components

Membrane
- Supports a component’s reflective capabilities
- Can support meta-object protocols through control interfaces
- Can have an internal structure of its own (cf AOKell)
- Arbitrary reflective capabilities can be supported (no fixed MOP)

Sharing
- A component can be a subcomponent of several composites
  - Component graphs, not just trees
- Useful for architectures with resources and cross-cutting concerns
A Fractal component

client interface

membrane

server interface

sub-component
Bindings

- Can support arbitrary communication semantics
  - request/response, asynchronous message dispatch, publish/subscribe, etc.
- Can be primitive or composite
  - A primitive binding connects two or more interfaces in the same address space; typically implemented by a language reference.
  - A composite binding connects two or more interfaces; can be reified as a component with primitive bindings.
- Can span address spaces and networks
Useful reflective capabilities (from the Fractal specification)

- **Reflection: minimal**
  - **Component controller** (discovering a component interfaces)
  - **Interface controller** (obtaining the Component controller)
  - **Binding controller** (binding an external component interface)

- **Reflection: structural**
  - **Content controller** (adding, removing subcomponents)
  - **Attribute controller** (setting, getting component attributes)

- **Reflection: behavioral**
  - **Interceptors** (before, around and after operations)
  - **Lifecycle controller** (starting, stopping the component)
Fractal: Forms of components

- Without reflection: objects
- With no sub-components and binding controller: objects with IOC
- With sub-components and binding controller: SCA components
- With component and interface controllers: COM components
- With binding and lifecycle controllers: OSGI bundles
- With binding controller and multicast bindings: CCA components
- With attribute controller: MBeans
- With transaction and persistency interceptors, controlling subcomponents with lifecycle controller: EJBeans
1. CBSE and Fractal

2. The Fractal Component Model

3. The Fractal Project

4. Fractal-based Developments

5. Conclusion
The Fractal Project

- An open source project
  - Hosted by the OW2 Open Source Middleware Consortium (ex ObjectWeb)
- That develops reflective software component technology for the construction of reconfigurable distributed systems
  - The Fractal component model
  - A set of tools for Fractal-based design, programming and deployment
Fractal Tools at http://fractal.ow2.org

**F4E**
- is an Eclipse based environment for developing Fractal applications.

**Fractal distribution**
- provides a ready to use package containing a collection of Fractal tutorials, and the whole libraries needed to develop Fractal applications.

**Fraclet**
- Java annotations for the Fractal component model.

**FScript**
- Domain-Specific Language (DSL) to program dynamic reconconfigurations of Fractal architectures.

**Fractal Explorer**
- Generic graphical console for managing Fractal-based applications. It allows to discover, introspect, manage, monitor and reconfigure Fractal applications at runtime.

**Fractal RMI**
- provides a set of components to create distributed bindings between Fractal components.

**Fractal ADL**
- is the base architecture description language of the Fractal component model. It provides a XML DTD for describing (for instance) component types, component implementations, component hierarchies and component bindings.

**Java Implementations**
- **Julia**
  - is the reference implementation of the Fractal component model. Using Julia it is possible to deploy a Fractal application relying on Julia runtime code generation of classes, interfaces, controllers. These tasks are accomplished internally using the ASM bytecode manipulation framework.
  - Julia
  - It allows generating and compiling the source code of the infrastructure (so called membranes) which is needed to run a Fractal/Java application. The application can then be run without having to use ASM to generate on-the-fly classes for interfaces, controllers and merge strategies.

**Fractal Java API**
- A language and message format that the implementations reify to provide interoperability between Java components and that Fractal programs use to communicate with the implementations.

**Fractal C API**
- is a development environment for programming Fractal components on the top of the C programming language.

**Main contributors:**
- INRIA
- France Telecom R&D
- STMicroelectronics

Jean-Bernard Stefani (INRIA Grenoble)
1. CBSE and Fractal

2. The Fractal Component Model

3. The Fractal Project

4. Fractal-based Developments

5. Conclusion
Operating system kernels: Think
Asynchronous middleware: Dream
Transaction management: GOTM
Deployment: DeployWare
Persistency services: Speedo, Perseus
Middleware for computational grids: Proactive
Middleware for enterprise application integration (EAI): Petals
Middleware for service oriented architectures (SOA): Frascati
Distributed systems management: Jade, Jasmine
Outline

1. CBSE and Fractal
2. The Fractal Component Model
3. The Fractal Project
4. Fractal-based Developments
5. Conclusion
Conclusion

- A component model for building reconfigurable software systems
  - Independent from programming languages
  - Open: reflection, composition, connection

- Supported by comprehensive series of open source tools
  - Lightweight implementations for component-based system programming
  - Run-time API for online reconfiguration
  - Combination of components and aspects
  - Programming and IDE support

- Used in industrial-strength developments
  - E.g. Speedo, Jasmine, Proactive, Petals

- Used in standards
  - Grid Component Model (GCM) standards at ETSI
What’s next?

- **Webinars:**
  - Fractal programming in Java
  - Fractal programming in C

- **Fractal technology:**
  - Formal semantics, design & verification
  - ADL for dynamic architectures
  - Workflow integration
  - Comprehensive deployment and configuration management support
To know more

- Visit the Fractal Web site: http://fractal.ow2.org
- Some publications: