FAC
Fractal Aspect Component

N. Pessemier, L. Seinturier, L. Duchien
T. Coupaye

INRIA Futurs, USTL-LIFL,
Jacquard project

www.objectweb.org
Get 2 development styles closer:

- Component assembly:
  - Packages software entities in modules
  - Increases reusability
  - Architecture Description Language (ADL):
    - Clarifies software entities interactions
  - Fractal component model

- Aspect-Oriented Programming (AOP):
  - Separation of Concerns (SoC)
  - Separates technical services from business logic
  - Modularizes crosscutting concerns
  - JAC model

- Fractal Aspect Component:
  - enhance Fractal component model and Fractal-ADL with AOP principles
Roadmap

- Fractal Crosscutting Issues

- FAC Abstract Model

- FAC Concrete Model

- Tools:
  - Fractal-ADL extension
  - Fractal Explorer extension

- Conclusion & Future Work
Fractal Crosscutting Issues
Crosscutting issues (1/3)
Crosscutting issues (2/3)

- Code scattering
- Code tangling

Diagram:

- ComponentA
- ComponentB
- ComponentC
- ComponentD
Crosscutting issues (3/3)

- Code scattering
- Code tangling

Apply AOP to components? Fractal Aspect Component
FAC Abstract Model
2 main notions:

- Aspect Component (AC): --> WHAT?
  - Embodies a crosscutting concern
  - Is a Fractal component with a server interface that describes the behavior to apply
  - Components and Aspects are components:
    - Symmetric approach

- Crosscutting Binding (CB): --> WHERE?
  - A new kind of binding between an aspect controller and an AC
  - New control interface: the aspect controller
An Aspect Component (AC) embodies a crosscutting concern.

Server interface AspectComponent

ComponentA

ComponentB

ComponentC

ComponentD
AC: Itf AspectComponent
AspectC: Aspect Controller
Abstract model (3/4)

AC: Itf AspectComponent
AspectC: Aspect Controller
CB: Crosscutting Binding
AC: Itf AspectComponent
AspectC: Aspect Controller
CB: Crosscutting Binding
3 levels to apply AOP:

- **Object:**
  - Content of components
  - Traditional AOP: AspectJ

- **Component:**
  - Aspect component (AC)
  - Crosscutting binding (CB)
  - Current FAC

- **Architecture:**
  - Architectural pattern transformation
  - Future work
FAC Concrete Model
FAC Concrete Model

- AspectComponent server interface
  - Defines an Aspect Component

- Aspect Control Interface
  - Supports the Crosscutting Bindings

- FAC Pointcut Language
  - Captures a set of components and interfaces
Aspect Component Interface

- An AC embodies a crosscutting concern
- ACs can be bound to business components

Server interface `AspectComponent`:
- Relies on AOP Alliance API: open source initiative to define a common API for AOP frameworks (Spring, Dynaop, Joyaop, JAC)
- Applies on client and server component interfaces
public class MyFirstAC implements AspectComponent {
    public Object invoke (MethodInvocation m) {
        // before method call
        Object ret = proceed();
        // after method call
        return ret;
    }
}
The Aspect Controller supports Crosscutting Bindings:
- Manages a list of ACs
- Allows local order policy on each component
- Relies on interception mechanism:
  - Julius (Julia extension): byte code transformation with ASM
  - FACAOKell (AOKell extension): AspectJ around advice.

BC : Binding Controller
LC : Life Cycle Controller
ACC : Aspect Component Controller
Aspect Controller API

Interface AspectComponent{
    // weaving part
    void weave(Component rootComp, AspectComponent ac, ItfPointcutExp pointcutExp, String cbName);
    void unweave(Component rootComp, Component ac);

    // Pointcut introspection part
    Component[] listAC();
    Component[] listCrosscutComps(Component rComp, Component ac);
    Pointcut aspectizableComps(Component rComp, ItfPointcutExp pExp);
}
Pointcut language: structural pointcuts

- Currently, 3 regular expressions declared in the ADL:
  - Component name
  - Interface name
  - Method signature
  - CLIENT/SERVER tag (type of interface)

- Pointcut examples:
  - ".* .* .*" – Intercepts every methods on every interfaces of all components
  - "SERVER server .* print.*:void" – Intercepts incoming void methods that begin with “print” in component “server”
Pointcut language: behavioral pointcuts

- The Aspect Component behavior is triggered on a sequence of messages (calls and executions of component interfaces)
- Historic/path of a method call
- A state machine is generated
- The expression describes its transitions

Example:
- Client.a();compA.b();compC.c() and OtherClient.d();compA.b();methodC.c()
- Wildcards authorized:
  - (a+b);c*;(d+e) matching sequences: ace, acd, bd, bce, …
<definition name="HelloWorld">
  <component name="client">
    <interface name="r" role="server"
      signature="java.lang.Runnable"/>
    <interface name="s" role="client"
      signature="Service"/>
    <content class="ClientImpl"/>
    <controller desc="primitive"/>
  </component>
  <component name="server">
    <interface name="s" role="server"
      signature="Service"/>
    <content class="ServerImpl"/>
    <controller desc="FACprimitive"/>
  </component>
  <weave
    root="this"
    ac="traceAC"
    componentExp="server"
    interfaceExp="s"
    methodExp=".*"
    pointcutName="traceCo"/>
</definition>
Fractal Explorer: introspection

List of ACs

ACs at the method granularity
Conclusions

- Weaving + Pointcut = new binding
- Symmetric approach: aspects & components are components
- 2 implementations: Julius & FACAOKell
- 3 levels:
  - Object
  - Component
  - Architecture
Perspectives: reach the architecture level

- Currently, advising of incoming and outgoing method calls on a component
- Get the pointcut language richer?
- Capture a binding?
- Capture a component that contains exactly 2 components?
Perspectives: reach the architecture level
Perspectives: reach the architecture level

Controller plan

Base plan
Perspectives: reach the architecture level

- The composite membrane is an independent composite
- Build a new component type is like adding a new composite in level M1
- Different policies:
  - One instance per component type
  - One instance per station
  - ...
- Advising at level M1:
  - Capture the bindings?
  - Capture the recomposition?