Recent Developments in AOKell

Fractal Workshop Middleware 2005

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

INRIA Lille

This work is partially funded by France Telecom
under the external research contract #46131097

Plan

1. Introduction
2. AOKell
3. AOKell-component
4. Dream with AOKell-component
5. FractNet

Introduction

AOKell
- implementation of the Fractal Specifications
- joint work with France Telecom R&D (T. Coupaye)

Implementations of the Fractal model (framework for engineering the control level)
- Julia: mixins & bytecode engineering (ASM)
- AOKell: aspects (AspectJ)

Expected benefits
- Easier to develop, debug, maintain new controllers
- Better integration with IDEs
- Reducing the development time for writing new controllers
- Reducing the learning curve

Introduction

AOKell

2 contributions
- aspect-oriented control level
- component-based control level

2 versions
- AOKell: aspects + objects
- AOKell-component: aspects + (control) components

AOKell

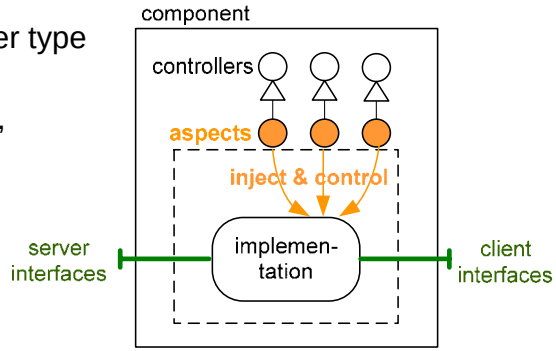
Requirements for controllers
- Feature injection (e.g. Binding controller interface)
- Interception (e.g. LifeCycle)

Our proposal
- 1 controller = 1 aspect
  - Feature injection = inter-type declaration (ITD)
  - Interception = code advising (before, after, around)

AspectJ
  - « reference » aspect weaver
  - Compile-time weaving (perf ++)
  - Load-time weaving (and run-time in the near future)
  - Tooling, IDE & debugger integration

AOKell

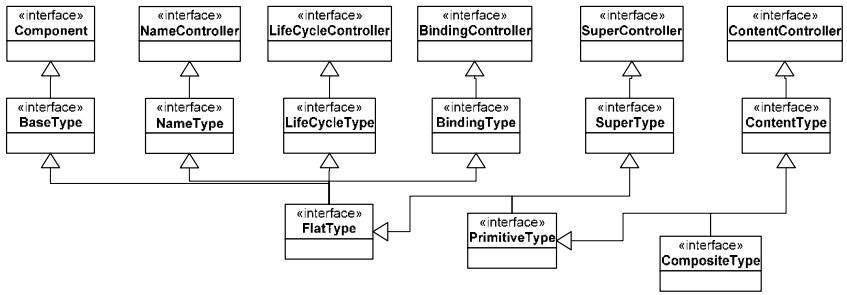
- 1 aspect per controller type
- 7 aspects
  BC, LC, NC, CC, SC,
  Factory, Component



- Each aspect delegates the control logic to a j.l.Object

AOKell

Controller « type system »



+ parametrics, templates and bootstrap

AOKell

```
public aspect ANameController {

   private NameController NameType._nc;

   public String NameType.getFcName() {
      return _nc.getFcName();
   }

   public void NameType.setFcName(String arg0) {
      _nc.setFcName(arg0);
   }

  public NameController NameType.getFcNameController() { return _nc; }
  public void NameType.setFcNameController(NameController nc) { _nc=nc; }
}
```

AspectJ
Inter-type declarations

Object implementation
of the name controller

AOKell

```
public aspect ALifeCycleController {

    private LifeCycleController LCType._lc;

    public String LCType.getFcState() { return _lc.getFcState(); }
    public void LCType.startFc() throws IllegalLifeCycleException { _lc.startFc(); }
    public void LCType.stopFc() throws IllegalLifeCycleException { _lc.stopFc(); }

    pointcut methodsUnderLifecycleControl( LCType adviced ):
        execution( * LCType+.*(..) ) && target(adviced) &&
        ! controllerMethodsExecution() && ! jlObjectMethodsExecution();

    before(LCType adviced) : methodsUnderLifecycleControl(adviced) {

      if( adviced.getFcState().equals(LifeCycleController.STOPPED) ) {
        throw new RuntimeException("Components must be started before
                                    accepting method calls");
```


AOKell

Full implementation of the Fractal specifications

- level 3.3
- API, ADL, template

Fractal/Julia junit conformance tests : 125 ok / total: 131
(6 failed: specific to Julia, or issues in interpreting the specs)

AOKell

Performances similar to those of Julia JACBenchmark

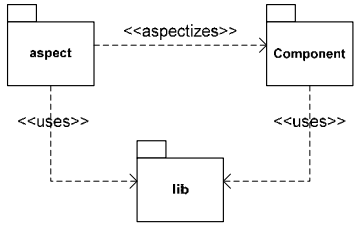
	Operation execution time	
	without interception	with interception
Pure Java 1.5.0	178ms	
AspectJ 1.2.1		209ms
Fractal/Julia 2.1.1	237ms	515ms
Fractal/AOKell 1.1	215ms	559ms
JBoss AOP 1.1.1		1046ms

AOKell

Project overview
.jar size: 187KB (152 + 35 for aspectjrt.jar) (Julia 2.2 180KB)

Applications tested with AOKell

- hw API, ADL, templates
- Fractal RMI
- Fractal Explorer
- comanche
- GoTM, cache-controller



Further works

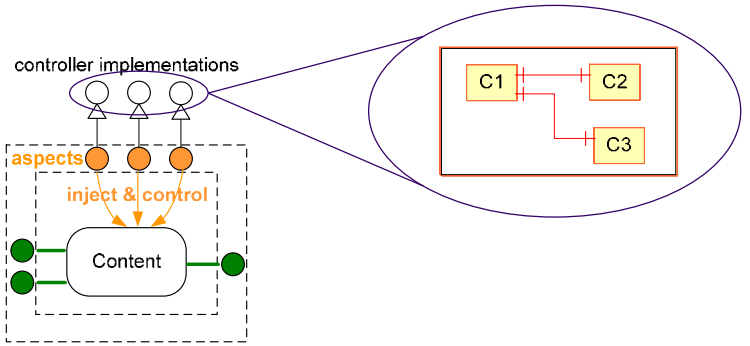
- Speedo, FROGi, …

Plan

1. Introduction
2. AOKell
3. AOKell-component
4. Dream with AOKell-component
5. FractNet

AOKell-component

Componentizing the membrane



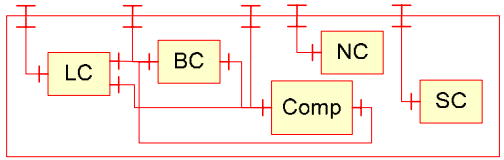
AOKell-component

Componentizing the membrane

- notion of a (primitive) component-controller
  - a component implementing a control interface
    BC, LC, NC, Component, SC,CC, Factory
- membrane
  - an assembly of components-controllers
  - a composite component-controller
  - can benefit from the tools available with Fractal
    e.g. Fractal-ADL

AOKell-component

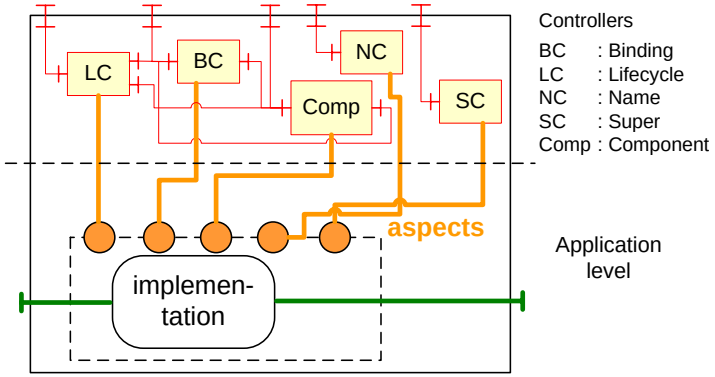
Example: membrane for a primitive component (1/3)



Controllers
BC : Binding
LC : Lifecycle
NC : Name
SC : Super
Comp : Component

AOKell-component

Example: membrane for a primitive component (2/3)

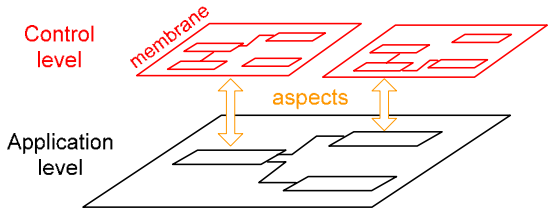


AOKell-component

```
<definition name="aokell.lib.membrane.primitive.Primitive"
      extends="LifeCycleType, BindingType, ComponentControllerType, NameControllerType,
      SuperControllerType" >

      <component name="Comp" definition="aokell.lib.control.component.PrimitiveComponentController" />
      <component name="NC"    definition="aokell.lib.control.name.NameController" />
      <component name="LC"    definition="aokell.lib.control.lifecycle.NonCompositeLifeCycleController" />
      <component name="BC"    definition="aokell.lib.control.binding.PrimitiveBindingController" />
      <component name="SC"    definition="aokell.lib.control.superc.SuperController" />

      <binding client="this.//component"          server="Comp.//component" />
      <binding client="this.//name-controller"    server="NC.//name-controller" />
      <binding client="this.//lifecycle-controller" server="LC.//lifecycle-controller" />
      <binding client="this.//binding-controller" server="BC.//binding-controller" />
      <binding client="this.//super-controller"   server="SC.//super-controller" />

      <binding client="Comp.//binding-controller" server="BC.//binding-controller" />
      <binding client="BC.//component«            server="Comp.//component" />
      <binding client="LC.//binding-controller"   server="BC.//binding-controller" />
      <binding client="LC.//component"            server="Comp.//component" />

      <controller desc="mComposite" />
</definition>
```


AOKell-component

Componentizing the membrane

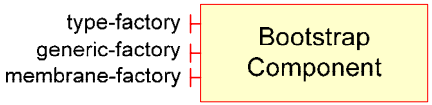


Control level
- no LC
- no templates
- Fractal level ~3.1 components
- 1 aspect per controller
- 1 root composite exporting the control interfaces

Application level
- Fractal level 3.3 components

AOKell-component

Fractal Bootstrap component



membrane-factory =
generic-factory supporting 2 new controller descriptions
mPrimitive and mComposite

Instantiating a component
- creating the content
- instantiating the membrane (composite component)
- linking the content and the membrane

AOKell-component

Issues: controlling the controllers?

Component controllers are regular Fractal components
⇨ they provide control interfaces (BC, NC, …)

How this (meta)-control level must be implemented?
1. controllers control themselves (meta-circularity)
2. ad-hoc implementation of the (meta)-control

AOKell-component

Conclusion

- Existing version of AOKell
  - 13 Fractal ADL membrane descriptions
  - Performances JACBenchmark
    additional cost wrt AOKell-OO: +20% (can certainly be reduced)
- A component-based engineering of the control
- 2 dimensions with AOKell
  - business: components
  - control: control components
  - glued together with aspects
- Perspectives: dynamic evolution of the control
  - need for (performant) run-time weaving

Plan

1. Introduction
2. AOKell
3. AOKell-component
4. Dream with AOKell-component
5. FractNet

AOKellized Dream

Dream

- a framework for developing middleware
- build on top of Fractal
- with new control membranes

- a version of Scalagent JORAM JMS server developed with Dream
- others … see INRIA Sardes project

AOKellized Dream

Dream & AOKell

- notion of a loggable component
- notion of an active component

- new controller interfaces
  - ContextualBindingController, BasicContentController, LocationController, LoggerController, LoggerControllerRegister, TaskController, TaskActivationController
- new membranes (9)
  - dreamPrimitive, activeDreamPrimitive, dreamUnstoppableComposite, …

AOKellized Dream

Implementing Dream with AOKell

- implement new controllers — Java
- implement aspects — AspectJ
- define membranes — Fractal-ADL

- register the membrane definitions with the AOKell kernel

AOKellized Dream

```
<definition name="aokell.dream.lib.membrane.ActiveStoppablePrimitive" extends="…" >

    <component name="ComponentController" definition="ComponentController" />
    <component name="LC" definition="ActiveFullLifeCycleController" />
    <component name="CBC" definition="PrimitiveContextualBindingController" />
    <component name="TC" definition="TaskController" />

      <!-- control interfaces exported omitted here -->

    <binding client="Comp.//binding-controller" server="CBC.//binding-controller" />
    <binding client="CBC.//component"           server="ComponentController.//component" />

    <binding client="LC.//component"                  server="ComponentController.//component" />
    <binding client="LC.//logger-controller"          server="LOGGERC.//logger-controller" />
    <binding client="LC.//task-controller"            server="TC.//task-controller" />
    <binding client="LC.//task-activation-controller" server="TC.//task-activation-controller" />

    <binding client="TC.//logger-controller"   server="LOGGERC.//logger-controller" />
    <binding client="TC.//binding-controller" server="CBC.//binding-controller" />
    <binding client="TC.//component"          server="ComponentController.//component" />

    <controller desc="mComposite" />
</definition>
```


AOKellized Dream

Implementing Dream with AOKell

- costliest task: implementing controllers

Proposed solution
- reuse existing controller implementations (Julia)
  - Julia controllers implemented with mixins
  - mixin class generator (*bytecode*)
- wrap them in AOKell control components

- a AOKell membrane can mix
  - Julia reused controllers
  - AOKell « natively » implemented controllers
- AOKellized Dream: lifecycle controllers reused

AOKellized Dream

Conclusion

- feasible
- hw is running ☺

- stress the implementation with real applications
- a transition from Julia to AOKell

Plan

1. Introduction
2. AOKell
3. AOKell-component
4. Dream with AOKell-component
5. FractNet

FractNet
an implementation of the Fractal Component Model for .NET

Clément ESCOFFIER, Didier DONSEZ
UJF/IMAG/LSR/ADELE
{clement.escoffier, didier.donsez}@imag.fr

.NET CLR
The other Virtual Machine

- Alternative to Java created/promoted by Microsoft
- Multi-language support
  - C# (ECMA/ISO), J#, VB.Net, Cobol.NET ...
- MSIL (MicroSoft Intermediate Language)
  - portable bytecode
- CLR : Virtual Machine for MSIL
- CLI : Runtime classes (partially ECMA+ISO)
- Several implementations
  - MS .NET (the official framework), Rotor (distributed by Microsoft for researchers and students)
  - Open Source : Mono (Linux), DotGNU (FreeBSD)

Component Model for .NET

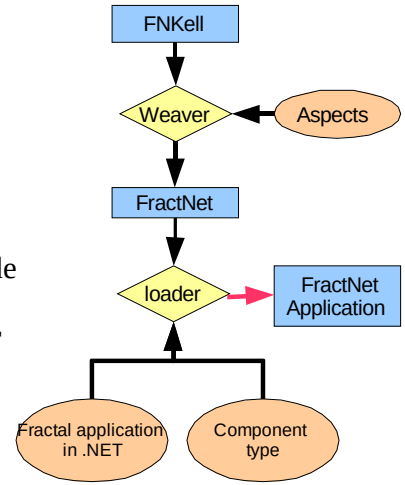
- Few general-purpose component models for .Net
  - Castle
    - A part of AVALON (fork in 2004-2005)
    - *Target mainly Web applications and Enterprise applications*
- Why not Fractal ?
  - Generic
  - Minimalist
  - Hierarchic
  - Extensible

FractNet : Motivations

- A Fractal implementation for .NET CLR
  - CLR supported languages (C#, J#, …)
- Based on the AOKell principles
  - Controllers are AspectJ aspects
- Use AspectDNG as aspect weaver
  - Multi-language weaver (J#, C#...)
  - Works on the MSIL code (a compiled assembly)
  - Can express AOKell's aspects
- Weaves aspects on a basic membrane, the FNKell
  - Provide this membrane for technical purpose
  - Can add/remove/modify controllers

FractNet: Generation suite

- Weaves aspects on FNKell
- Use AOKell to generate membrane
- Hint : Currently
  - Decompile the Java bytecode of the generated classes
  - Them recompile it for .NET with J# compiler
  - The generated code is « directly » compliant with J#



FractNet architecture

Performances (JAC Benchmark)

For 10.000.000 loops*	Minimum time (in ms)	Ratio FractNet/x	Mean time (in ms)	Ratio FractNet/x
Pure Java	2380	1,426	2386	1,496
Pure .Net	2854	1,190	2886	1,237
FractNet	**3395**	**1,000**	**3569**	**1,000**
Julia Merge All	5556	0,611	5558	0,642
AOKell	6339	0,536	6359	0,561
Julia Normal	6889	0,493	6892	0,518

* Configuration Intel Centrino 1,7 GHz, 512MB RAM, Windows XP SP 2.
SUN JavaVM 1.5.0 Client vs MS .NET CLR 2.0.50215

FractNet : the perspectives

- Current status : proof of concept !
  - Available on http://www-adele.imag.fr/fractnet
- Short term
  - Runtime generation of component type and bindings
  - Caching for generated classes (use of the .NET Global Assembly Cache)
- Mid-Term
  - FractNetADL: FractalADL for .NET
  - FractNetRemoting
    - method invocation on a remote FN Component using .NET Remoting (ORPC, HTTP/SOAP, IIOP)
    - Interoperability with Java Fractal implementations
  - Dynamic Service-Oriented Component for OSGi.NET
    - Same FROGi's goal
- No Roadmap
- Contributors (may be sponsors) are welcome
  - Lionel, Nicolas, Vladimir, … ?