Fraclet: Making Fractal Developments Easier!

Wednesday, January 25th, 2006

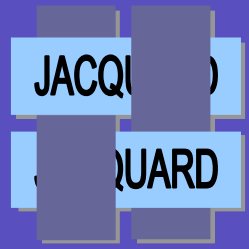
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

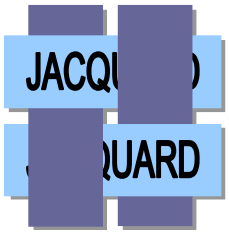
INRIA

1st Internal Jacquard Workshop ;-)

Romain **Rouvoy**, *Philippe* **Merle**

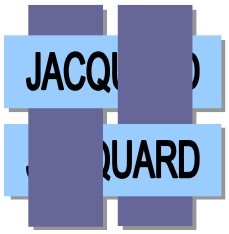
Email: *romain.rouvoy@inria.fr*
INRIA Futurs – Jacquard Project,
Laboratoire d'Informatique Fondamentale de Lille, France





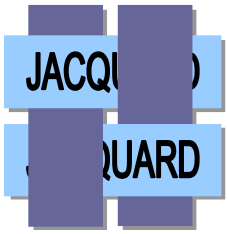
Agenda

- Motivations
- Contributions
- Conclusion
- Perspectives



Problem

- The Fractal component model is an interesting component model:
  - Various implementations: *Julia, AOKell, ProActive, etc*.
  - Various extensions: *FAC*, etc.
  - Various tools: *FractalADL, FractalGUI, FractalRMI, FractalExplorer, etc*.
  - Various applications: *Dream, GoTM, etc*.
  - Availability: http://fractal.objectweb.org/
- BUT, the Fractal component model may quickly become boring:
  - Implementation of the component business code,
  - Definition of an AttributeController per Component,
  - Implementation of Fractal controller callbacks:
    - BindingController: to support client (required) interfaces,
    - LifeCycleController: to be notified of start/stop actions,
    - XXXAttributeController: to support attribute (re)configuration.
  - Definition of the component ADL descriptor.



Problem

- The Fractal component model is an interesting component model:
  - Various implementations: *Julia, AOKell, ProActive, etc*.
  - Various extensions: *FAC*, etc.
  - Various tools: *FractalADL, FractalGUI, FractalExplorer, etc*.
  - Various applications: *Dream, GoTM, etc*.
  - Availability: http://fractal.objectweb.org/

- BUT, the Fractal component model may quickly become boring:
  - Implementation of the component business code,
  - Definition of an AttributeController interface,
  - Implementation of Fractal controller callbacks:
    - BindingController: to support client (required) interfaces,
    - LifeCycleController: to be notified of start/stop actions,
    - XXXAttributeController: to support attribute (re)configuration.
  - Definition of the component ADL descriptor.

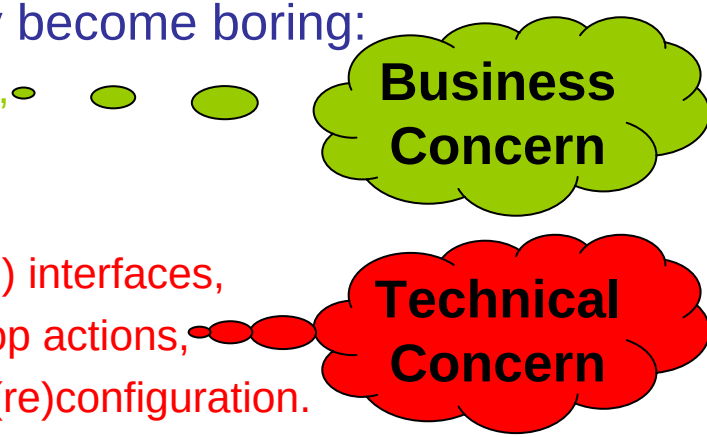
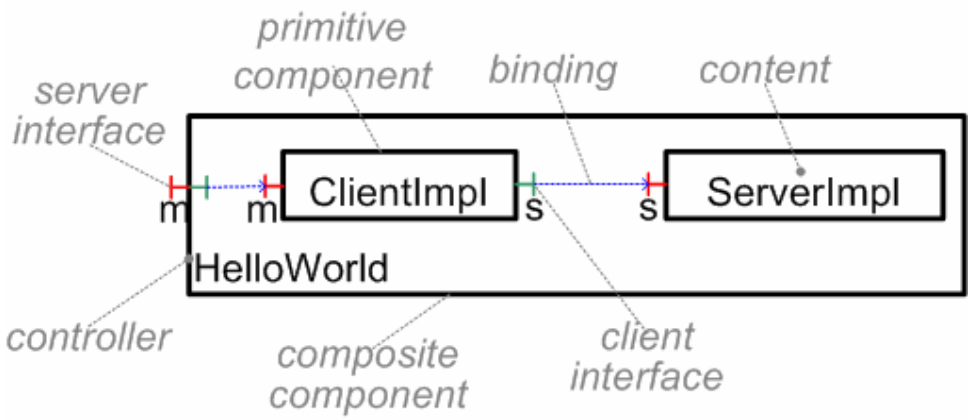
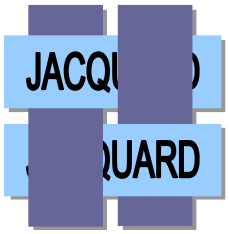


Illustration with the HelloWorld Example





HelloWorld: Java interfaces (from Julia)

- **Client business interface:**

```
public interface Main {
        void main (String[] args);
}
```

- **Server business interface:**

```
public interface Service {
        void print (String msg);
}
```

- **Server attribute configuration interface:**

```
public interface ServiceAttributes extends AttributeController {
        String getHeader ();
        void setHeader (String header);
        int getCount ();
        void setCount (int count);
}
```

HelloWorld: Java interfaces (from Julia)

- **Client business interface:**

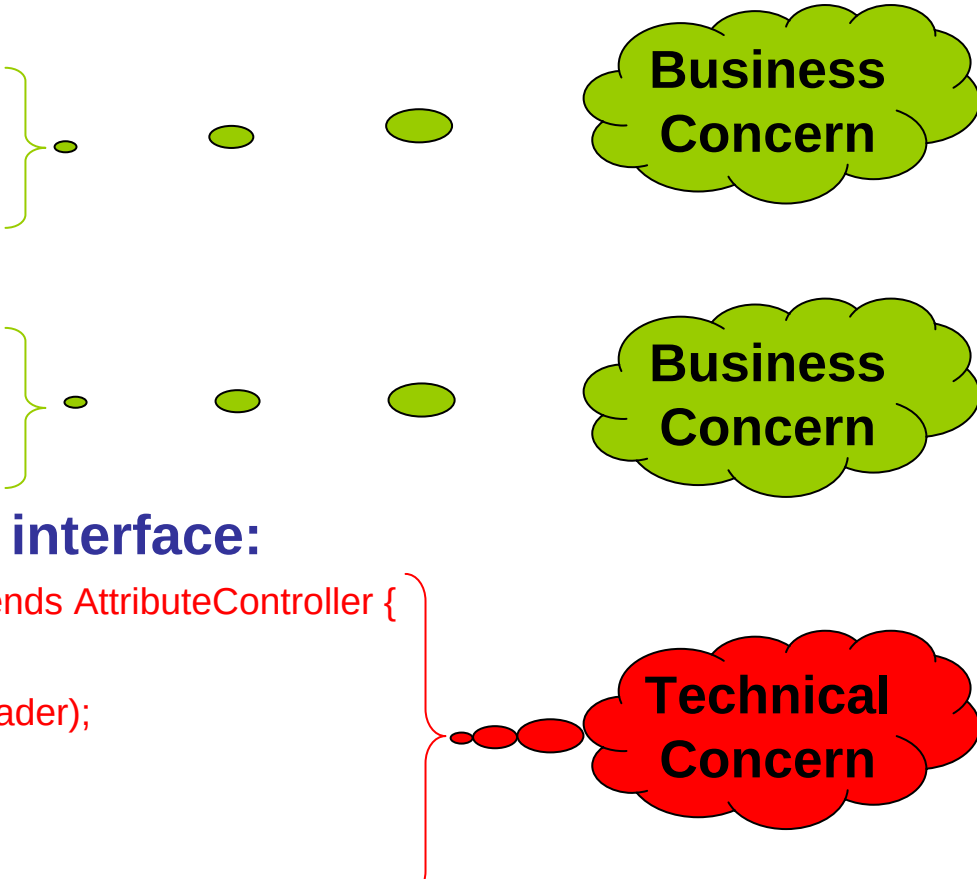
```
public interface Main {
        void main (String[] args);
}
```

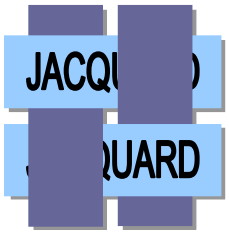
- **Server business interface:**

```
public interface Service {
        void print (String msg);
}
```

- **Server attribute configuration interface:**

```
public interface ServiceAttributes extends AttributeController {
        String getHeader ();
        void setHeader (String header);
        int getCount ();
        void setCount (int count);
}
```



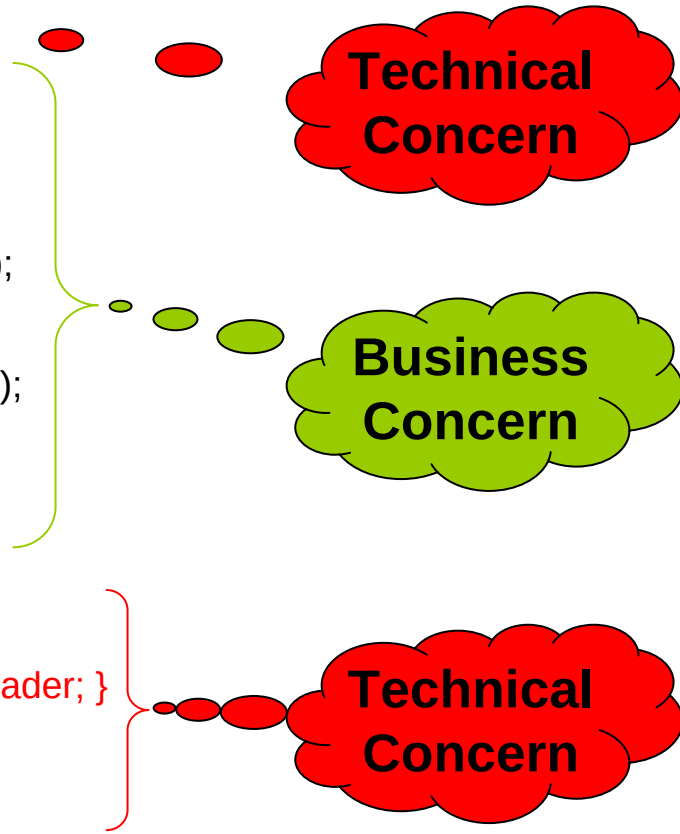


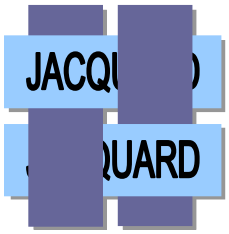
HelloWorld: Server code (from Julia)

```
public class ServerImpl implements Service, ServiceAttributes {
        private String header = "";
        private int count = 0;
        public void print (final String msg) {
                System.err.println("Server: begin printing...");
                for (int i = 0; i < count; ++i) {
                        System.err.println(header + msg);
                }
                System.err.println("Server: print done.");
        }
        public String getHeader () { return header; }
        public void setHeader (String header) { this.header = header; }
        public int getCount () { return count; }
        public void setCount (int count) { this.count = count; }
}
```

HelloWorld: Server code (from Julia)

```
public class ServerImpl implements Service, ServiceAttributes {
        private String header = "";
        private int count = 0;
        public void print (final String msg) {
                System.err.println("Server: begin printing...");
                for (int i = 0; i < count; ++i) {
                        System.err.println(header + msg);
                }
                System.err.println("Server: print done.");
        }
        public String getHeader () { return header; }
        public void setHeader (String header) { this.header = header; }
        public int getCount () { return count; }
        public void setCount (int count) { this.count = count; }
}
```



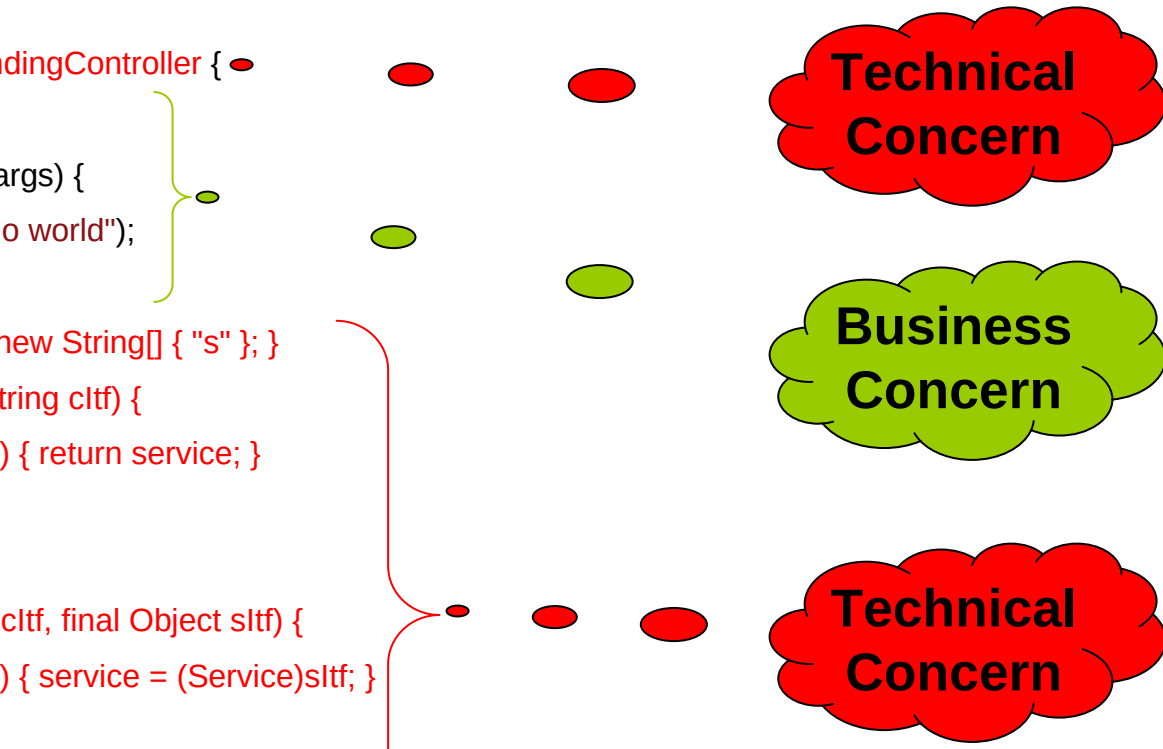


HelloWorld: Client code (from Julia)

```
public class ClientImpl implements Main, BindingController {
        private Service service;
        public void main (final String[] args) {
                service.print("hello world");
        }
        public String[] listFc () { return new String[] { "s" }; }
        public Object lookupFc (final String cItf) {
                if (cItf.equals("s")) { return service; }
                return null;
        }
        public void bindFc (final String cItf, final Object sItf) {
                if (cItf.equals("s")) { service = (Service)sItf; }
        }
        public void unbindFc (final String cItf) {
                if (cItf.equals("s")) { service = null;    }
        }
}
```

HelloWorld: Client code (from Julia)

```
public class ClientImpl implements Main, BindingController {
        private Service service;
        public void main (final String[] args) {
                service.print("hello world");
        }
        public String[] listFc () { return new String[] { "s" }; }
        public Object lookupFc (final String cItf) {
                if (cItf.equals("s")) { return service; }
                return null;
        }
        public void bindFc (final String cItf, final Object sItf) {
                if (cItf.equals("s")) { service = (Service)sItf; }
        }
        public void unbindFc (final String cItf) {
                if (cItf.equals("s")) { service = null;    }
        }
}
```



HelloWorld: Server ADL definition (from fractaladl)

- **Definition of component's type (optional):**

```
<definition name="ServerType">
    <interface name="s" role="server" signature="Service"/>
</definition>
```

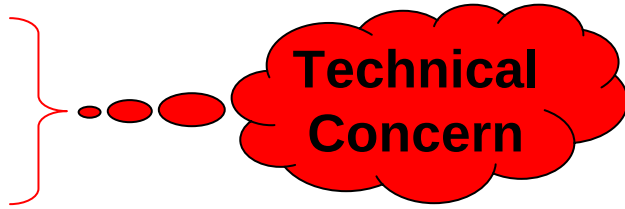
- **Definition of component's implementation:**

```
<definition name="ServerImpl" extends="ServerType">
    <content class="ServerImpl"/>
    <attributes signature="ServiceAttributes">
        <attribute name="header" value= ">> "/>
        <attribute name="count" value="1"/>
    </attributes>
    <controller desc="primitive"/>
</definition>
```

HelloWorld: Server ADL definition (from fractaladl)

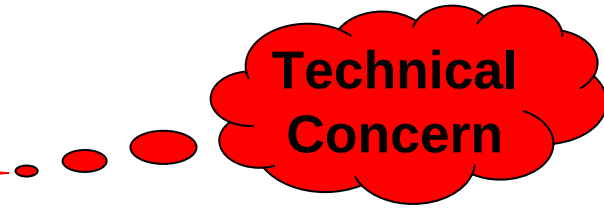
- **Definition of component's type (optional):**

```
<definition name="ServerType">
    <interface name="s" role="server" signature="Service"/>
</definition>
```



- **Definition of component's implementation:**

```
<definition name="ServerImpl" extends="ServerType">
    <content class="ServerImpl"/>
    <attributes signature="ServiceAttributes">
        <attribute name="header" value= ">> "/>
        <attribute name="count" value="1"/>
    </attributes>
    <controller desc="primitive"/>
</definition>
```



HelloWorld: Client ADL definition (from fractaladl)

- **Definition of component's type (optional):**

```
<definition name="RootType">
    <interface name=”m" role="server" signature=”Main"/>
</definition>

<definition name="ClientType" extends="RootType">
    <interface name="s" role="client" signature="Service"/>
</definition>
```

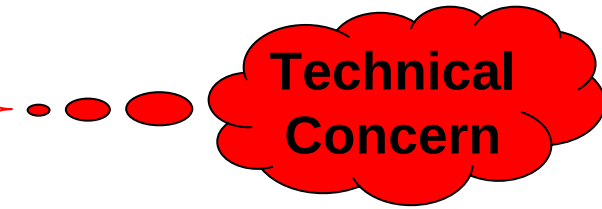
- **Definition of component's implementation:**

```
<definition name="ClientImpl" extends="ClientType">
    <content class="ClientImpl"/>
</definition>
```

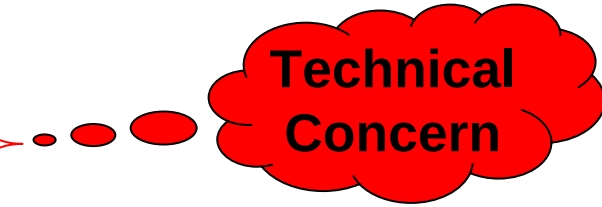
HelloWorld: Client ADL definition (from fractaladl)

- **Definition of component's type (optional):**

```
<definition name="RootType">
    <interface name=”m" role="server" signature=”Main"/>
</definition>
```

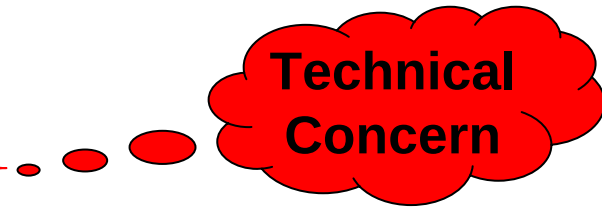



```
<definition name="ClientType" extends="RootType">
    <interface name="s" role="client" signature="Service"/>
</definition>
```



- **Definition of component's implementation:**

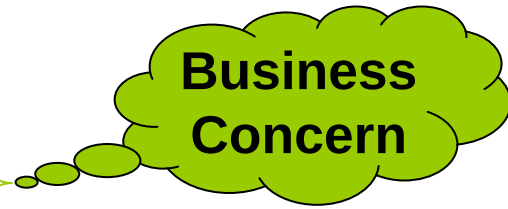
```
<definition name="ClientImpl" extends="ClientType">
    <content class="ClientImpl"/>
</definition>
```



HelloWorld: HelloWorld ADL definition (from fractaladl)

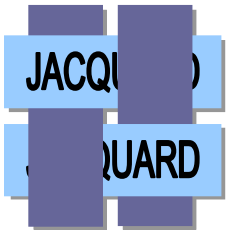
- **Definition of component's type (optional):**

```
<definition name="AbstractClientServer" extends="RootType">
    <component name="client" definition="ClientType"/>
    <component name="server" definition="ServerType"/>
    <binding client="this.m" server="client.m"/>
    <binding client="client.s" server="server.s"/>
</definition>
```



- **Definition of component's implementation:**

```
<definition name="ClientServerImpl" extends="AbstractClientServer">
    <component name="client" definition="ClientImpl"/>
    <component name="server" definition="ServerImpl"/>
</definition>
```

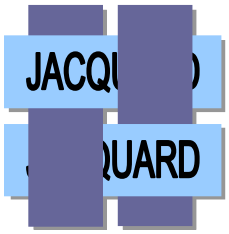


Synthesis

- 24 lines of code are related to a business concern:
  - *4 lines to define the business interfaces,*
  - *7 lines to implement the server component,*
  - *3 lines to implement the client component,*
  - *10 lines to describe the assembly.*

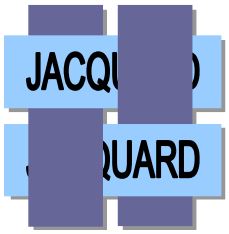
- 37 lines of code are related to a technical concern:
  - *9 lines are related to the AttributeController callback,*
  - *8 lines are related to the BindingController callback,*
  - *20 lines are related to the ADL definition of primitive components.*

- ~60% of the code written by the developer is non-functional!
  - *Repetitive task, subject to errors,*
  - *Introduce dependencies towards the middleware (i.e. controller interfaces),*
  - *Time-consuming extensions (e.g. add an attribute).*



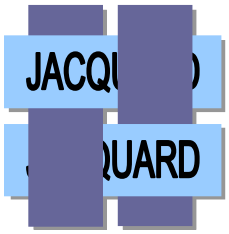
Code Generators: Existing tools

- **Eclipse IDE** (Java to Java)
  - *Plus: Native support for template generators (getter/setter templates),*
  - *Minus: Limited to static Java code generation (no support for ADL),*
  - *Minus: Tightly coupled to the IDE,*
  - *Minus: No incremental support.*
- **FractalGUI Tool** (ADL to Java)
  - *Plus: Graphical design of Fractal-based applications,*
  - *Plus: Java code template generator,*
  - *Minus: No reverse engineering capabilities,*
  - *Minus: No support for legacy components.*
- **Object-oriented Development** (Java)
  - *Plus: Use of Design patterns,*
  - *Plus: Factorization of technical code:*
    - *e.g., Generic binding controller.*
  - *Minus: Overhead introduced:*
    - *Abstract method to implement,*
    - *Execution overhead due to delegation principle.*
  - *Minus: Limited to Java / No support for ADL generation.*



Challenges

- Existing approaches are not sufficient:
  - *Incomplete approaches (incomplete template class / no ADL generation),*
  - *Bad support for code evolution (adding a new attribute),*
  - *Bad integration with legacy components (developments from scratch).*
- Need for an incremental technical code generator:
  - *Full support of the Fractal component model,*
  - *Full support of the Fractal ADL tool,*
  - *Support for code evolution,*
  - *Seamlessness tool for developers.*
- Definition of an annotation framework: Fraclet
  - *Introduction of metadata in the Java code,*
  - *Generation of various artifacts from the Java code,*
  - *Complementary approach of Java code,*
  - *Compliant with existing IDEs (from Eclipse to Vi),*
  - *Use XDoclet2 as annotation & generation framework (http://xdoclet.codehaus.org/)*



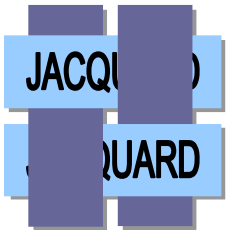
The XDoclet philosophy

•XDoclet will parse your source files and generate many artifacts such as XML descriptors and/or source code from it. These files are generated from templates that use the information provided in the source code and its **JavaDoc tags**.

•XDoclet lets you apply **Continuous Integration** in component-oriented development. Developers should concentrate their editing work on **only one Java source file per component**.

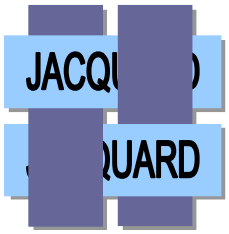
•Java Open Source Programming:

- *Currently XDoclet can only be used as part of the build process utilizing Jakarta Maven or Ant.*
- *Although XDoclet originated as a tool for creating EJBs, it has evolved into a **general-purpose code generation engine**. XDoclet consists of a core and a constantly growing number of modules. It is fairly straightforward to write new modules if there is a need for a new kind of component.*
- *XDoclet comes with a set of modules for generation of different kinds of files. Users and contributors can write their own modules (or modify existing ones) if they wish to extend the functionality of XDoclet.*



The XDoclet benefits

- This approach has several benefits:
  - *You don't have to worry about out dating deployment meta-data whenever you touch the code. The deployment **meta-data is continuously integrated**.*
  - *Working with only one file per component gives you a **better overview of what you're doing**. If your component consists of several files, it's easy to lose track. If you have ever written an Enterprise Java Bean, you know what we mean. A single EJB can typically consists of 7 or more files. With XDoclet you only maintain one of them, and the rest is generated.*
  - *You **dramatically reduce development time**, and can concentrate on business logic, while XDoclet generates 85% of the code for you.*

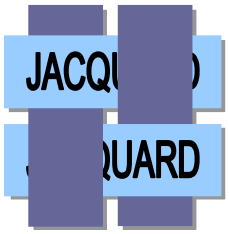


Fraclet: an XDoclet2 plugin for Fractal

- Objective:
  - *Extension of XDoclet2 tool to support the Fractal Component Model,*
  - *Generation of Fractal components glue (i.e. controllers callbacks),*
  - *Generation of FractalADL descriptors (for primitive components).*

- Requirements:
  - *Definition of Fractal annotations,*
  - *Definition of templates for artifacts generation.*

- Extra-Requirements:
  - *Reuse of Fractal concepts (seamlessness property),*
  - *Maximize as much as possible the size/number of generated artifacts,*
  - *Reduce as much as possible the verbosity of annotations.*

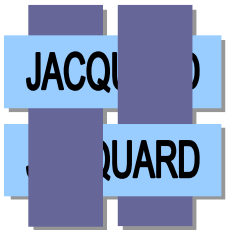


Fraclet Overview

- Definition of 3 class annotations:
  - *@fractal.itf, @fractal.ctrl, @fractal.tmpl*
  - *Devoted to the generation of FractalADL descriptors.*

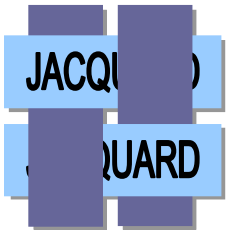
- Definition of 4 field annotations:
  - *@fractal.bc, @fractal.ac, @fractal.rc, @fractal.log*
  - *Devoted to the generation of Component glue & FractalADL descriptors.*

- Definition of 5 XDoclet2 plugins:
  - *Generation of extended AttributeController interface,*
  - *Generation of Fractal component glue,*
  - *Generation of FractalADL prmitive descriptors,*
  - *Generation of FractalADL composite descriptors,*
  - *Generation of Monolog configuration file.*



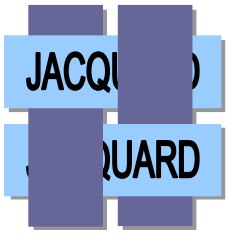
@fractal.itf class annotation

- Describes a Fractal interface.
- Syntax: @fractal.itf name="…" signature="…"
- Description:
  - **name** [required]: *name of the interface*,
  - **signature** [optional]: *default value is class signature*,
- Example:
  - /** @fractal.itf name="run" */
    public interface Runnable { …}



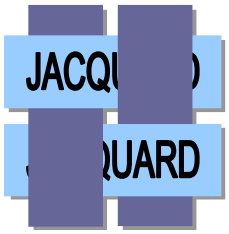
@fractal.ctrl class annotation

- Describes the controller part of a Fractal component.
- Syntax: @fractal.ctrl desc="…"
- Description:
  - **desc** [required]: *descriptor of the controller part*.
- Example:
  - /** @fractal.ctrl desc="my-primitive" */
    public class Request { …}



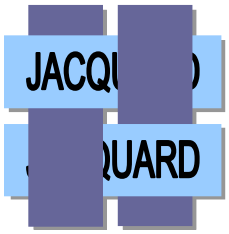
@fractal.tmpl class annotation

- Describes the template controller part of a Fractal component.
- Syntax: @fractal.tmpl desc="…"
- Description:
  - **desc** [required]: *descriptor of the template controller part*.
- Example:
  - /** @fractal.tmpl desc="my-primitive-template" */
    public class Request { …}



@fractal.bc field annotation

- Describes the BindingController of a component.
- Syntax: @fractal.bc name="…" signature="…" contingency="…" cardinality="..."
- Description:
  - **name** [optional]: *default value is the name of the field*,
  - **signature** [optional]: *default value is the signature of the field*,
  - **contingency** [mandatory|optional]: *default value is* mandatory,
  - **cardinality** [singleton|collection]: *default value is* singleton.
- Example:
  - /** @fractal.bc */
    protected Runnable run; // field need to be defined as protected.

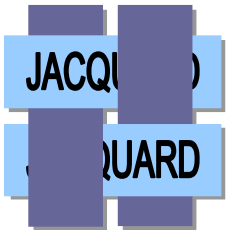


@fractal.ac field annotation

- Describes the AttributeController of a component.
- Syntax: @fractal.ac name="…" value="…" argument="…"
- Description:
  - **name** [optional]: *default is the name of the field*,
  - **value** [optional]: *default is the signature of the field*,
  - **argument** [optional]: *default is* client,

  If neither value nor argument is specified then a default argument parameter is declared with the field name.

- Example:
  - /** @fractal.ac */
    protected String header; // field need to be defined as protected.

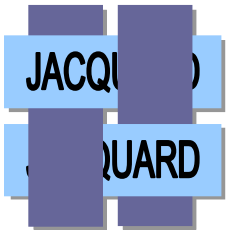


@fractal.rc field annotation

- Provides a reference to the controllers of the component.
- Syntax: @fractal.rc controller="…"
- Description:
  - **controller** [optional]: default is "component".

- Example:
  - /** @fractal.rc */
    protected Component self; // field need to be defined as protected.

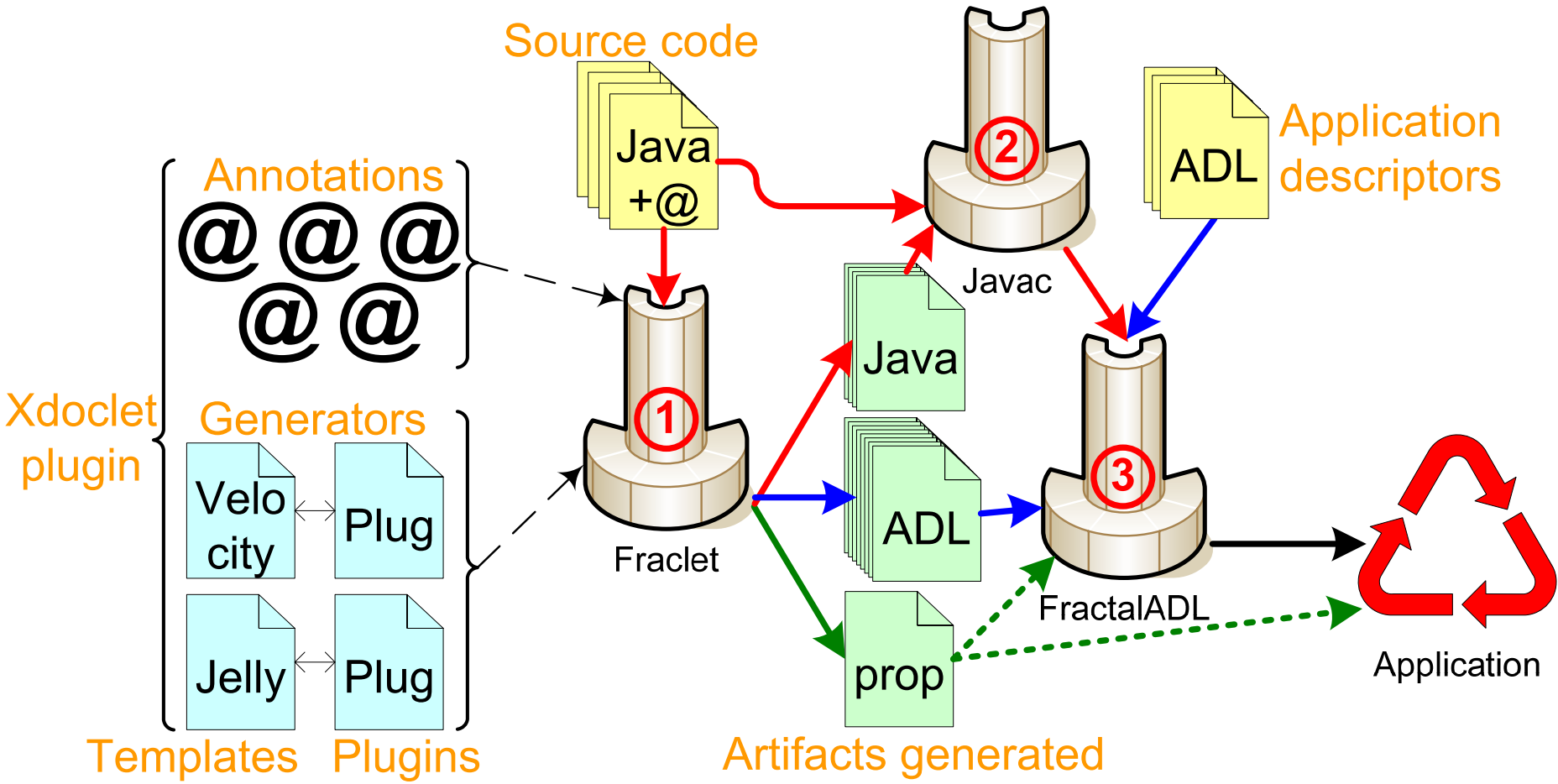
self references the component interface of the current component.



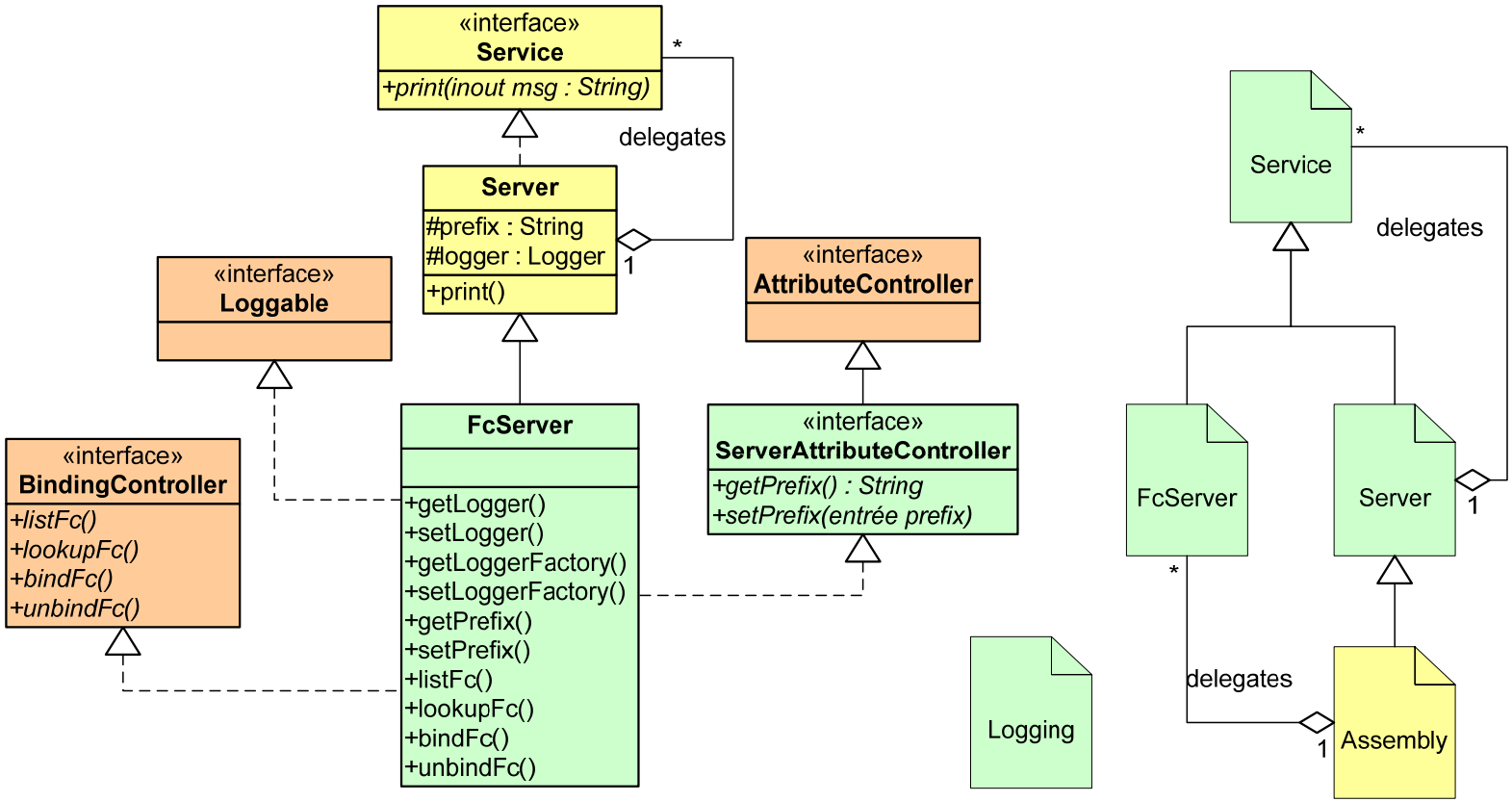
@fractal.log field annotation

- Initializes a logger for the component.
- Syntax: @fractal.log name="…" level="…" handler="…" additivity="…" clean-handlers="..."
- Description:
  - **name** [optional]: *default value is the full name of the component*,
  - level [optional]: *default value is* INFO,
  - handler [optional]: *default value is* consoleHandler,
  - additivity [optional]: *default value is* true,
  - clean-handlers [optional]: *default value is* true.
- Example:
  - /** @fractal.log */
    protected Logger log; // field need to be defined as protected.

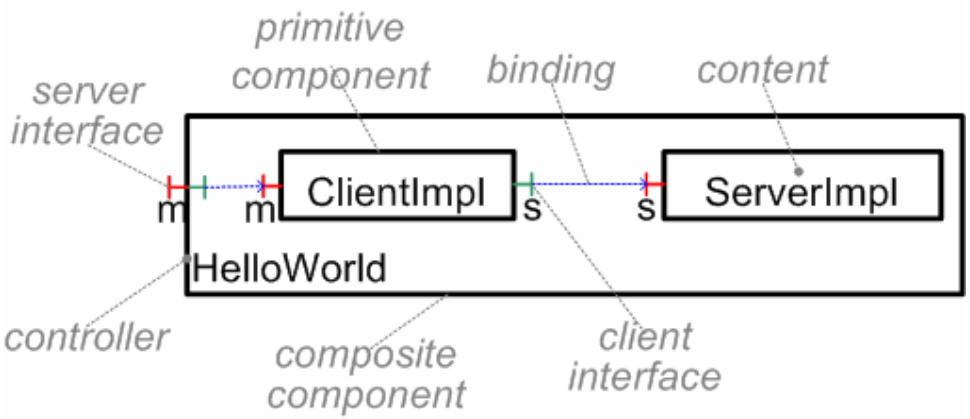
Fraclet Generation Process

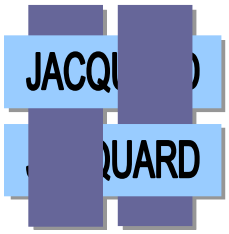


Generated Class Inheritance Tree



Revisiting HelloWorld from Julia





HelloWorld: Java interfaces (from Julia)

- **Client business interface:**

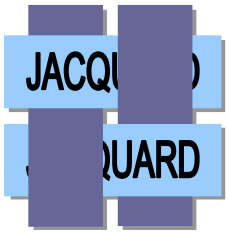
```
public interface Main {
        void main (String[] args);
}
```

- **Server business interface:**

```
public interface Service {
        void print (String msg);
}
```

- **Server attribute configuration interface:**

```
public interface ServiceAttributes extends AttributeController {
        String getHeader ();
        void setHeader (String header);
        int getCount ();
        void setCount (int count);
}
```



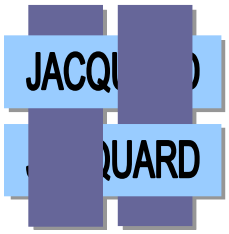
HelloWorld: Java interfaces (with Fraclet)

- Client business interface:

```
/** @fractal.itf name="m" */
public interface Main {
        void main (String[] args);
}
```

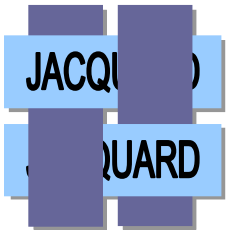
- Server business interface:

```
/** @fractal.itf name="s" */
public interface Service {
        void print (String msg);
}
```



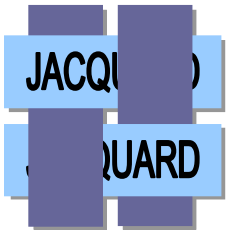
HelloWorld: Server code (from Julia)

```
public class ServerImpl implements Service, ServiceAttributes {
        private String header = "";
        private int count = 0;
        public void print (final String msg) {
                System.err.println("Server: begin printing...");
                for (int i = 0; i < count; ++i) {
                        System.err.println(header + msg);
                }
                System.err.println("Server: print done.");
        }
        public String getHeader () { return header; }
        public void setHeader (String header) { this.header = header; }
        public int getCount () { return count; }
        public void setCount (int count) { this.count = count; }
}
```



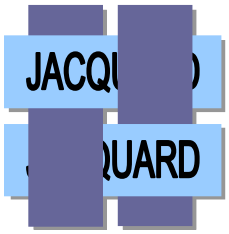
HelloWorld: Server code (with Fraclet)

```
public class ServerImpl implements Service{
        /** @fractal.ac */
        protected String header = "";

        /** @fractal.ac */
        protected int count = 0;

        public void print (final String msg) {
                System.err.println("Server: begin printing...");
                for (int i = 0; i < count; ++i) {
                        System.err.println(header + msg);
                }
                System.err.println("Server: print done.");
        }
}
```



HelloWorld: Client code (from Julia)

```
public class ClientImpl implements Main, BindingController {
        private Service service;
        public void main (final String[] args) {
                service.print("hello world");
        }
        public String[] listFc () { return new String[] { "s" }; }
        public Object lookupFc (final String cItf) {
                if (cItf.equals("s")) { return service; }
                return null;
        }
        public void bindFc (final String cItf, final Object sItf) {
                if (cItf.equals("s")) { service = (Service)sItf; }
        }
        public void unbindFc (final String cItf) {
                if (cItf.equals("s")) { service = null;    }
        }
}
```



HelloWorld: Client code (with Fraclet)

```
public class ClientImpl implements Main {

        /** @fractal.bc name="s" */
        protected Service service;

        public void main (final String[] args) {

                service.print("hello world");

        }
}
```

HelloWorld: Server ADL definition (from fractaladl)

- **Definition of component's type (optional):**

```
<definition name="ServerType">
  <interface name="s" role="server" signature="Service"/>
</definition>
```

- **Definition of component's implementation:**

```
<definition name="ServerImpl" extends="ServerType">
  <content class="ServerImpl"/>
  <attributes signature="ServiceAttributes">
    <attribute name="header" value= ">> "/>
    <attribute name="count" value="1"/>
  </attributes>
  <controller desc="primitive"/>
</definition>
```

HelloWorld: Server ADL definition (with Fraclet)

- The definitions are automatically generated:
  - **FcService.fractal** describes the Service interface,
  - **FcServerImpl.fractal** describes the concrete ServerImpl component,

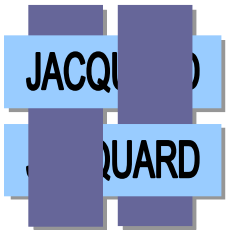
HelloWorld: Client ADL definition (from fractaladl)

- **Definition of component's type (optional):**

```
<definition name="RootType">
    <interface name=”m" role="server" signature=”Main"/>
</definition>

<definition name="ClientType" extends="RootType">
    <interface name="s" role="client" signature="Service"/>
</definition>
```

- **Definition of component's implementation:**

```
<definition name="ClientImpl" extends="ClientType">
    <content class="ClientImpl"/>
</definition>
```



HelloWorld: Client ADL definition (with Fraclet)

- The definitions are automatically generated:
  - **FcMain.fractal** describes the Main interface,
  - **FcClientImpl.fractal** describes the primitive component ClientImpl,
  - **ClientImpl.fractal** describes the abstract composite component containing the primitive component FcClientImpl.

HelloWorld: HelloWorld ADL definition (from fractaladl)

- **Definition of component's type (optional):**

```
<definition name="AbstractClientServer" extends="RootType">
   <component name="client" definition="ClientType"/>
   <component name="server" definition="ServerType"/>
   <binding client="this.m" server="client.m"/>
   <binding client="client.s" server="server.s"/>
</definition>
```

- **Definition of component's implementation:**

```
<definition name="ClientServerImpl" extends="AbstractClientServer">
   <component name="client" definition="ClientImpl"/>
   <component name="server" definition="ServerImpl"/>
</definition>
```

HelloWorld: HelloWorld ADL definition (with Fraclet)

- **Definition of component's type (optional):**

```
<definition name="AbstractClientServer" extends="Client">
   <component name="client" definition="Client"/>
   <component name="server" definition="Server"/>
   <binding client="this.m" server="client.m"/>
   <binding client="client.s" server="server.s"/>
</definition>
```

- **Definition of component's implementation:**

```
<definition name="ClientServerImpl" extends="ClientImpl">
   <component name="server" definition="FcServerImpl('>> ',1)"/>
</definition>
```

Evaluation on Comanche

	Julia (A)	Fraclet (B)	*Xdoclet*	Gain (G=A-B)	**Rate (G/A)**
Java files	12	12	*3*	0	**0%**
Fractal files	19	6	*10*	13	**68%**
Java sources	254	173	*152*	81	**32%**
Fractal sources	88	38	*30*	50	**57%**
Source Size	124K	72K	*68K*	52K	**41%**
Bytecode Size	124K	*144K*		-20K	**-14%**

- Comanche uses the binding controller in 3 primitive components.
- Comanche do not use any attribute controller.
- The structure of the final assembly is preserved (even if details change).

Conclusion – Our Contribution

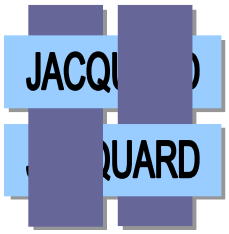
- An annotation framework to leverage the development of applications with Fractal
  - *Higher level of abstraction to describe meta-data of components,*
  - *No dependency towards Fractal libraries,*
  - *Reduction of Java code.*

- An implementation of the framework using the XDoclet2 tool
  - *Full support of Fractal specificities using 5 annotations,*
  - *3 generators :Attribute Controller, Component Glue, ADL Descriptor,*
  - *Annotations are described using Javadoc tags,*
  - *Generators executed from the Ant or Maven build process.*

- Integrated in the **Fractal project** (http://fractal.objectweb.org/)
  - Available via the ObjectWeb Maven repository,
  - *Compliant with Julia & AOKell (http://aokell.gforge.inria.fr/),*
  - *Project in progress, implementation details can change …*

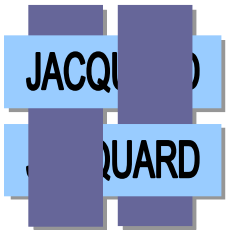
Our Future Work

- Definition of annotations for the FractalExplorer tool:
  - *Generation of XML configuration files,*
  - *Generation of explorer actions.*
- Generation of graphical representation of Fractal components:
  - *Parsing FractalADL descriptors,*
  - *Generating SVG graphical representations.*
- Extended Support for FractalRMI, FractalJAR, etc.
- Towards Java 1.5 annotations:
  - *Implementation of Fraclet using Java annotations,*
  - *Using Spoon as Generation engine (http://spoon.gforge.inria.fr/).*
- Integration in the FractalGUI tool to support reverse engineering
- Documentation



Thank you for your attention…

If you have any questions?

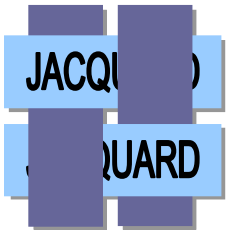


The FractalADL Template (1/3)

- Based on the Jelly XML generation engine.

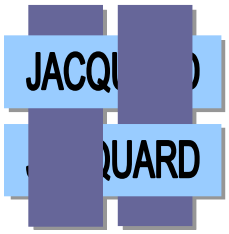
•Description:

```
<j:jelly xmlns:j="jelly:core" xmlns:x="jelly:xml">

<x:doctype name="definition"
      publicId="-//objectweb.org//DTD Fractal ADL 2.0//EN"
      systemId="classpath://org/objectweb/fractal/adl/xml/standard.dtd"/>

<x:comment text="${dontedit}" trim="true"/>
<definition name="${plugin.qualifiedDefinition(metadata)}"
        extends="${plugin.extendDefinition(metadata)}"
      arguments="${plugin.argumentDefinition(metadata)}">
```



The FractalADL Template (2/3)

```
<j:forEach var="tag" items="${metadata.getTagsByName('fractal.itf')}">
  <interface name="${tag.getName_()}"
             role="${tag.getRole()}"
               contingency="${tag.getContingency()}"
       cardinality="${tag.getCardinality()}"
                 signature="${plugin.getSignature(metadata,tag)}"/>
</j:forEach>

<j:if test="${plugin.isPrimitive(metadata)}">
  <content class="${metadata.getFullyQualifiedName()}"/>
</j:if>
```



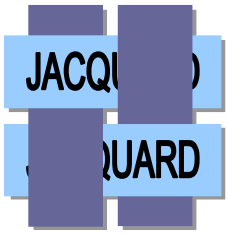
The FractalADL Template (3/3)

```
<j:if test="${plugin.hasAttributeController(metadata)}">
  <attributes signature="${plugin.attributeController(metadata)}">
  <j:forEach var="tag" items="${metadata.getTagsByName('fractal.attr')}">
    <attribute name="${tag.getName_()}"
value="${plugin.attributeValue(tag)}"/>
  </j:forEach>
  </attributes>
</j:if>

<j:if test="${metadata.getTagByName('fractal.ctrl')!=null}">
  <controller desc="${metadata.getTagByName('fractal.ctrl').getDesc()}"/>
</j:if>
```

The AttributeController template (1/1)

- Based on the Velocity code generation engine.

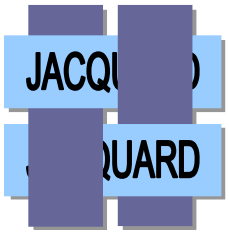
```
public interface ${plugin.definition($class)} extends attributecontroller {
#foreach ( $field in $plugin.attributes($class) )
#set( $name = $plugin.attributemethod($field) )
    /**
     * setter for the ${name} attribute.
     * @param value value for the ${name} attribute.
     */
    void set${name}(${field.gettype()} value);

    /**
     * getter for the ${name} attribute.
     * @return value of the ${name} attribute.
     */
    ${field.gettype()} get${name}();
#end
}
```



The component glue template (1/4)

- Based on the Velocity code generation engine.

```
/**
#foreach ( $attribute in ${attributes} )
#set( $doclet = $attribute.getTagByName('fractal.ac') )
#if ( ${doclet.getValue_()} )
 * @fractal.attr name="${plugin.attributeName($attribute)}" value="${doclet.getValue_()}"
#elseif ( ${plugin.attributeArgument($attribute)} )
 * @fractal.attr name="${plugin.attributeName($attribute)}"
                 argument="${plugin.attributeArgument($attribute)}"
#end
#end
#foreach ( $binding in ${bindings} )
#set( $doclet = $binding.getTagByName('fractal.bc') )
#if ( !(${doclet.getHidden()}) )
 * @fractal.itf name="${plugin.bindingName($binding)}" role="client"
cardinality="${doclet.getCardinality()}" contingency="${doclet.getContingency()}"
signature="${plugin.bindingSignature($binding)}"
#end
#end
 */bbbb
```

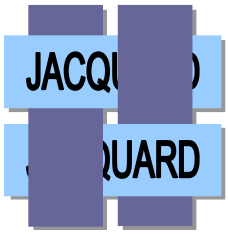


The component glue template (2/4)

```
public class ${plugin.definition($metadata)} extends ${class.name}
#if(${plugin.hasAttributes($class)} && ${plugin.hasBindings($class)})
  implements ${class.name}AttributeController, BindingController
#elseif ( ${plugin.hasAttributes($class)} )
  implements ${class.name}AttributeController
#elseif ( ${plugin.hasBindings($class)}  )
  implements BindingController
#end
{
```

The component glue template (3/4)

```
#foreach ( $field in ${attributes} )
#set( $name = ${plugin.attributeMethod($field)} )
    /**
     * Setter for the ${name} attribute.
     * @param value value for the ${name} attribute.
     */
    public void set${name}(final ${field.getType()} value) {
        super.${field.getName()} = value ;
    }

    /**
     * Getter for the ${name} attribute.
     * @return value of the ${name} attribute.
     */
    public ${field.getType()} get${name}() {
        return super.${field.getName()} ;
    }
#endbb
```



The component glue template (4/4)

```
#if ( ${plugin.hasBindings($class)}  )
    private final ArrayList fcBindings = new ArrayList();

    public String[] listFc() {
        return (String[]) this.fcBindings.toArray(new String[this.fcBindings.size()]);
    }

    public Object lookupFc(final String name) throws NoSuchInterfaceException {
#foreach ( $binding in ${bindings} )
        if(name.startsWith("${plugin.bindingName($binding)}"))
#if ( $binding.getTagByName('fractal.bc').getCardinality().equals("singleton") )
            return this.${binding.getName()} ;
#else
            return this.${binding.getName()}.get(name);
#end
        else
#end
#if ( ${class.isA('org.objectweb.fractal.api.control.BindingController')} )
            return super.lookupFc(name);
#else
            throw new NoSuchInterfaceException("Client interface \'"+name+"\' is undefined.");
#end
    }
```