

Jonathan & Fractal

Fractal Workshop
29 January 2003

Eric Bruneton



➔ Jonathan « Fractalization »

- First experiment
- Second experiment

➔ Fractal RMI

➔ Future work

First experiment

➔ Instantiate Jonathan with Julia instead of Kilim 1

- No source code modification
- Defined a generic KilimTemplate class
 - to instantiate Kilim components with Fractal templates
- Translated Kernel.kilim file into Kernel.fractal

➔ Results

- Showed that Kilim and Fractal templates are similar
- Performances overhead
 - Null without interceptors (static configuration)
 - 7% with interceptors everywhere (fully reconfigurable configuration)

Second experiment

➔ Towards a true « fractalization » of Jonathan

- Source code modifications
 - Implement UserBindingController and AttributeController
- Attempted to define a hierarchical architecture
 - Jonathan's kilim configuration file described a « flat » architecture

➔ Results

- Fractalization was quite easy
 - because Jonathan was already a component based software!
 - some problems remain (« map » interfaces)
- Found a uniform architecture for all personalities
 - much more understandable than the original, flat description

Second experiment

```
public class A implements I {
    J comp;
    int attr;
    public A (J comp, int attr) {
        this.comp = comp;
        this.attr = attr;
    }
    ...
}
```



```
public interface AAttributes extends AttributeController {
    int getAttr ();
    void setAttr (int attr);
}

public class A implements I, UserBindingController, AAttributes {
    J comp;
    int attr;
    public A () { }
    public A (J comp, int attr) {
        this.comp = comp;
        this.attr = attr;
    }
    public int getAttr () { return attr; }
    public void setAttr (int attribute) { this.attr = attr; }
    public void getFcBindings (String itf) { ... }
    public void addFcBinding (String itf, Object obj { ... }
    public void removeFcBinding (String itf, Object obj) { ... }
    ...
}
```

➔ New, lightweight personality of Jonathan

- Synchronous remote method calls
- `InterfaceReference` replaces `java.rmi.Remote`
- `RemoteException` is *not* required
- No stub compiler tool
 - stub and skeleton classes are dynamically generated with ASM
- 5 new components, 6 Jonathan components reused
- 150K instead of 2M for David or Jeremie

➔ Used to create distributed Fractal applications

➔ Will be put into Fractal's code base soon

Future work

- ➔ **Fix some fractalization problems**
 - add « map » interfaces (in addition to single and collection)?
- ➔ **Fractalize protocol components**
- ➔ **Evaluate advantages of the fractalization**
 - Dynamic reconfigurations?
- ➔ **Measure performances (impact of Julia)**
- ➔ **Integrate changes into Jonathan's code base**
- ➔ **Reify distributed bindings as Fractal components?**
 - Is it useful? How much does it cost?