Making SOFA Fractal level 2 compliant

Petr Hnětynka, František Plášil

DISTRIBUTED SYSTEMS RESEARCH GROUP
http://nenya.ms.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics
Agenda

• SOFA overview
• Fractal conformance levels
• Making SOFA compliant with Fractal
  ▪ level 1
  ▪ level 2
• Results
SOFA overview

- SOFA
  - Software Appliances
- component model
  - DCUP
    - Dynamic Component Updating
- communication using connectors
- behavior specification
- component distribution/delivery
  - SOFA node(s)
- multiplatform
  - prototype in Java
- http://sofa.objectweb.org/
Development process

• CDL spec done by hand
• Compiling CDL done automatically
  ▪ Check Protocol Compliance
• Generating Code fragments
  ▪ Types
  ▪ Component builders
  ▪ Assembly descriptors
• Primitive components
• Assembling
  ▪ Filling out Assembly descr
• Deploying application to Depl. Docks
  ▪ Filling out Deployment plan
  ▪ Connector generation
• Launching
Implementation of primitive component

- CM
- CB
- implementation objects
Implementation of composed component
Fractal conformance levels

- Fractal – 4 levels of conformance
  0. nothing is mandatory
  1. components provide **Component** interface
  2. interfaces are castable to **Interface** interface
  3. components use *type system*

- SOFA conforms on level 0.
- but
  - SOFA and Fractal are very similar
  → higher level of conformance
Making SOFA compliant with Fractal

• level 1. and 2.
  ▪ easy
  ▪ providing **Component** interface and extending **Interface** interface
    • necessary – mapping from Fractal IDL to SOFA CDL

• level 3.
  ▪ problematic
  ▪ type system – definition of sub typing relation
  ▪ SOFA sub typing defined via behavior protocols
Mapping: Fractal IDL $\rightarrow$ SOFA CDL

- straightforward
  - interface $\rightarrow$ interface
  - exception $\rightarrow$ exception
  - package $\rightarrow$ module
  - primitive types $\rightarrow$ primitive types
  - unbound array $\rightarrow$ unbound sequence
package org.objectweb.fractal.api;

interface Component {
    any[] getFcInterfaces();
    any getFcInterface(string itfName)
        throws NoSuchInterfaceException;
    Type getFcType();
}

interface Type {
    boolean isFcSubtypeOf(Type t);
}
module org {
    module objectweb {
        module fractal {
            module api {

                interface Type {
                    boolean isFcSubtypeOf(in Type t);
                };

                typedef sequence<Object> Objects;

                interface Component {
                    Objects getFcInterfaces();
                    Object getFcInterface(in string itfName)
                        raises (NoSuchInterfaceException);
                    Type getFcType();
                };
            };
        };
    };
}
Fractal *Interface* interface

```java
package org.objectweb.fractal.api;

interface Interface {
    string getFcItfName();
    Type getFcItfType();
    Component getFcItfOwner();
    boolean isFcInternalItf();
}
```
module org {
    module objectweb {
        module fractal {
            module api {

                interface Interface {
                    string getFcItfName();
                    Type getFcItfType();
                    Component getFcItfOwner();
                    boolean isFcInternalItf();
                }
            }
        }
    }
}

Petr Hnětynka, František Plášil
Fractal workshop, March 25, 2004, Paris
Level 1 conformance

- implemented
- Component manager implements interface `Component`
  - also implements `Type` interface
- `getFcInterfaces`, `getFcInterface`
  - returns connector role (role ~ connector ending)
    - pre-generated connectors
    - no on-the-fly generation
- minor change in mapping from CDL to Java
  - each generated interface/class contains static and final field with CDL full name
    - used for obtaining correct connector
- `isFcSubtypeOf()`
  - currently returns only `false`
Level 2 conformance

- implemented
- modified mapping from CDL to Java
  - all interfaces implicitly extends `Interface` interface
- modified connector generator
  - adding of methods from `Interface` interface
  - special handling of method `getFcltfOwner`
    - returns reference to Component manager
- problem
  - method `isFtInternalIft()`
    - no internal interface in SOFA
    - each time returns `false`
Achieved results

- SOFA conforms to Fractal on level 2
  - easier building heterogeneous component applications
- component introspection
- making component manager publicly accessible
  - dynamic architecture of components