

Fractal and management

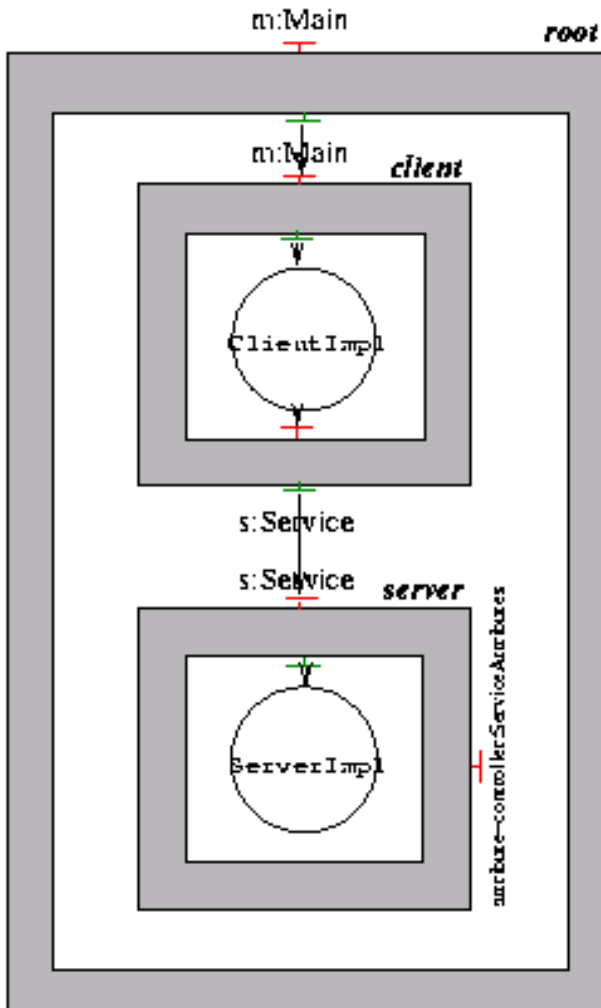
What could be done in Fractal related
to application management ?

L. Andrey, Madynes project, Loria/Inria-Lorraine, 29/01/03
andrey@loria.fr

1st ideas and example

- Instrumentation/monitoring
 - configuration: some real work is done somewhere else.
 - "applications are the resources"... ?
- Provide easy link with "legacy" administration *frameworks*...
 - Jmx: main target ?
 - for Ow project refactoring stuff using fractal.
 - I.e j2ee JSR
- Stay as much as possible at Fractal level
 - ⇒ use assembly time to introduce "instrumentation" of functional components

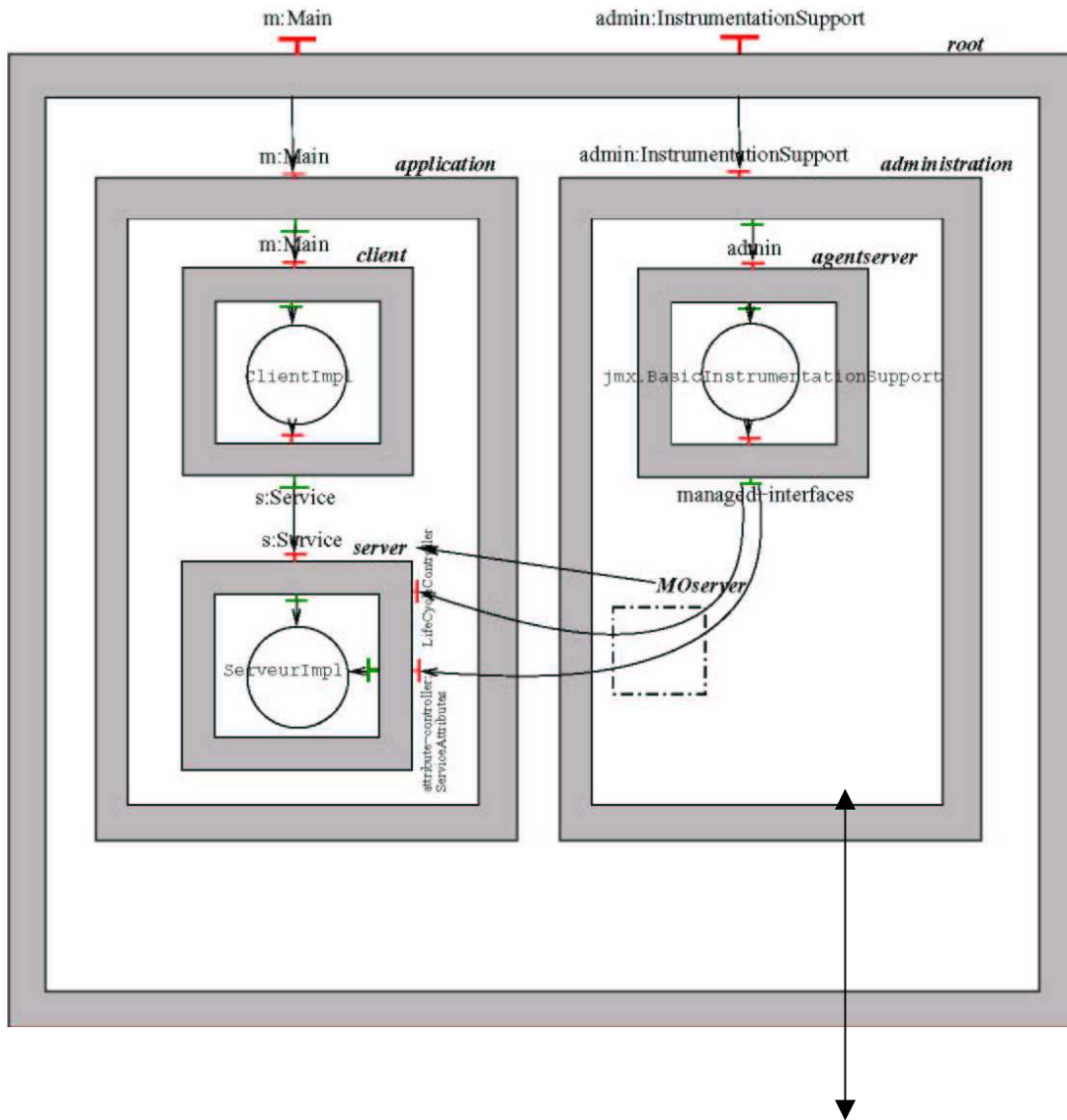
Hello World



```
<primitive-component name="client">
  <provides>
    <interface name="m" type="Main"/>
  </provides>
  <requires>
    <interface name="s" type="Service"/>
  </requires>
  <class name="ClientImpl"/>
</primitive-component>
<primitive-component name="server">
  <provides>
    <interface name="s" type="Service"/>
    <interface name="attribute-controller"
      type="ServiceAttributes"/>
  </provides>
  <requires/>
  <class name="ServerImpl"/>
  <attributes>
    <attribute name="Header" value="-> "/>
    <attribute name="Count" value="1"/>
  </attributes>
</primitive-component>
</subcomponents>
<bindings>
  <binding client="application.m" server="client.m"/>
  <binding client="client.s" server="server.s"/>
</bindings>
```

Rmk: default controller interfaces are not depicted (LifeCycle, ...)

Hello World + jmx access



- fractal service instances, associated to a name (jmx) are exposed
- a fractal component does the exposition (*agentserver*)
- bindings between "*managed-interface*" and interfaces to expose
- instrumentation components are kept apart

Jmx access

Hello World + JMX (2)

```
<composite-component name="administration">
  <provides>
    <interface name="admin" type="jmx.InstrumentationSupport"/>
  </provides>
  <requires/>
  <subcomponents>
    <primitive-component name="agentserver">
      <provides>
        <interface name="admin" type="jmx.InstrumentationSupport"/>
      </provides>
      <requires>
        <interface name="managed-interfaces" type="" sort="collection"/>
      </requires>
      <class name="jmx.BasicInstrumentationSupport"/>
    </primitive-component>
    <shared-component name="M0server">
      <path value="root.application.server"/>
    </shared-component>
  </subcomponents>
  <bindings>...
  <binding client="agentserver.managed-interfaces0" server="M0server.lifecycle-controller"/>
  <binding client="agentserver.managed-interfaces1" server="M0server.attribute-controller"/>
  <binding client="agentserver.managed-interfaces2" server="agentserver.lifecycle-controller"/>
  <binding client="agentserver.managed-interfaces3" server="agentserver.admin"/>
  </bindings>
</composite-component>
```

Jmx Mapping

- Fractal *attributes-controller* interface -> set of JMX attributes
- Other fractal interface -> set of JMX operations
- Name...a JMX name must be provided... somehow...
 - built from name of "shared" component or *name-controller*, plus default JMX domain and key (fractal attributes of the "*agentserver*")
 - defined by user... extra xml stuff (todo).

JMX Mapping (2)

- Current test uses ModelMBean
 - lots of JMX descriptor fields are not used... How user can specify then ? Extra xml stuff for binding ? Todo.
- (Stupid) restriction: one generated ModelMBean only maps fractal interfaces from the same ComponentIdentity owner.
 - Why ? Because I am lazy...
 - Any needs to do otherwise ? Yes, management view may want to group and expose attributes in a different way than Component level.
 - This kind of special views can be specified with extra fractal Components (in the *Administration* composite), and then exposed them as previously shown...
- Problem: method name slashes. Fractal interfaces method names are flattened in the created ModelMBean scope.

What next ?

- Fractalization of JMX...
 - Connectors, adaptors => configure the *agentserver*
 - JMX component like *monitors* => specify some monitoring functions at the fractal level
- Better test implementation (easy ;->)
 - un/re/binding to *management-interfaces* => update Mbeans (exposed attributes or operations, registration)
 - Gauge, counter threshold by dynamic code generation ?
- Mapping Component models... General pb...