Fractal ADL v2

E. Bruneton

Overview

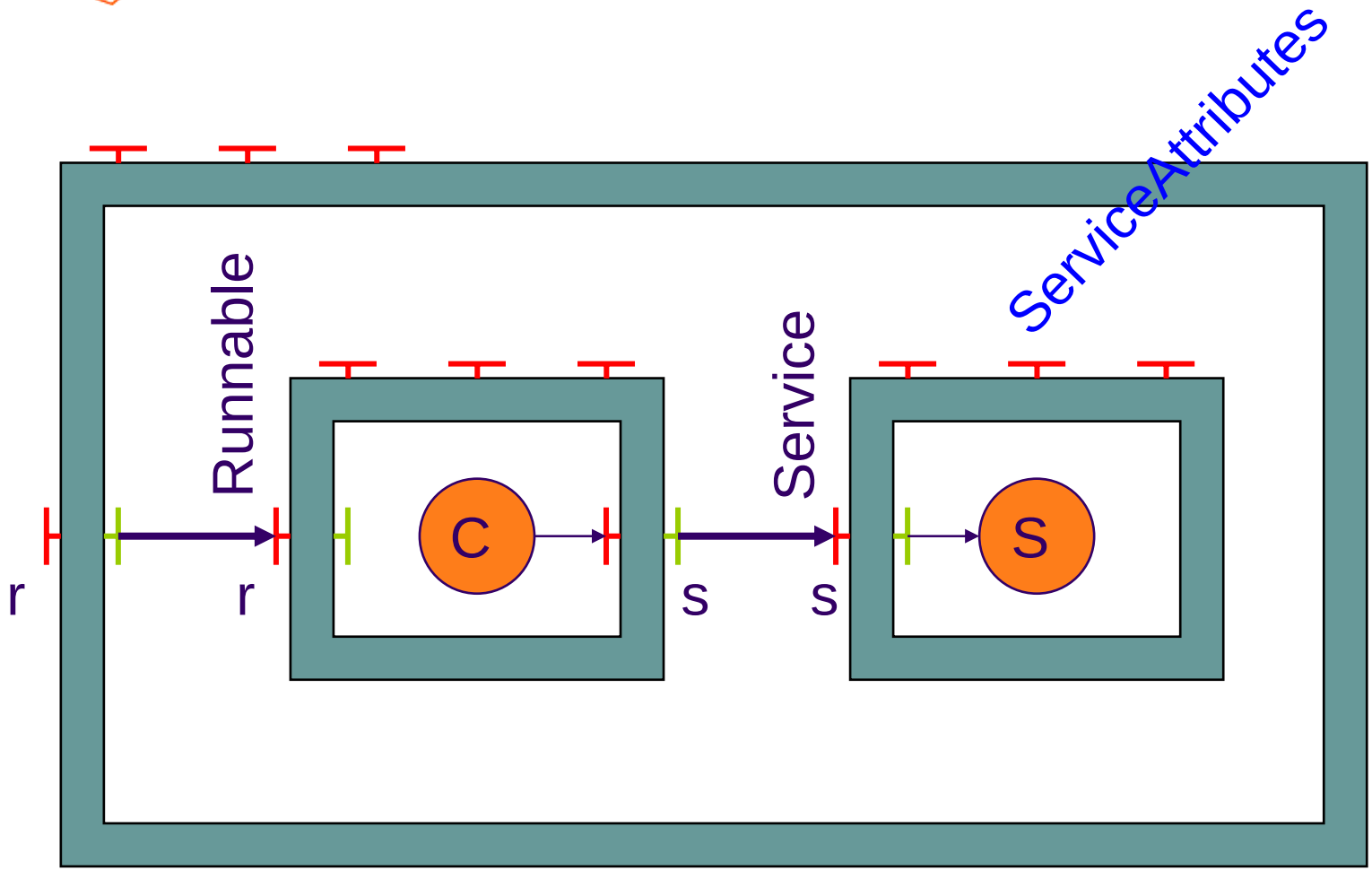
- Introduction
- Primitive components
- Composite components
- Component controllers
- Component references
- Arguments
- Nodes
- Usage

Introduction

Fractal ADL

- A language to define component architectures for Fractal,
- An open and extensible language:
  - made of a set of *modules* (each module defines an *abstract syntax*)
    - Interfaces,
    - Bindings,
    - Attributes,
    - Sub components,
    - Implementation,
    - Virtual nodes,
    - …

HelloWorld example



Primitive components

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <!DOCTYPE definition PUBLIC
     "-//objectweb.org//DTD Fractal ADL 2.0//EN"
     "classpath://org/objectweb/fractal/adl/xml/basic.dtd">

 <definition name="ClientImpl">
   <interface name="r" role="server" signature="java.lang.Runnable"/>
   <interface name="s" role="client" signature="Service"/>
   <content class="ClientImpl"/>
 </definition>
```

Composite components

```
<definition name="HelloWorld">
  <interface name="r" role="server" signature="java.lang.Runnable"/>
  <component name="client">
    <interface name="r" role="server" signature="java.lang.Runnable"/>
    <interface name="s" role="client" signature="Service"/>
    <content class="ClientImpl"/>
  </component>
  <component name="server">
    <interface name="s" role="server" signature="Service"/>
    <content class="ServerImpl"/>
  </component>
  <binding client="this.r" server="client.r"/>
  <binding client="client.s" server="server.s"/>
</definition>
```

Component controllers

```
<definition name="ServerImpl">
  <interface name="s" role="server" signature="Service"/>
  <content class="ServerImpl"/>
  <attributes signature="ServiceAttributes">
    <attribute name="header" value="->"/>
    <attribute name="count" value="1"/>
  </attributes>
  <controller desc="primitive"/>
</definition>
```

Component references (1)

```
<definition name="HelloWorld">
  <interface name="r" role="server" signature="java.lang.Runnable"/>
  <component name="client" definition="ClientImpl"/>
  <component name="server">
    <interface name="s" role="server" signature="Service"/>
    <content class="ServerImpl"/>
  </component>
  <binding client="this.r" server="client.r"/>
  <binding client="client.s" server="server.s"/>
</definition>
```

Component references (2)

```
<definition name="ClientType">
  <interface name="r" role="server" signature="java.lang.Runnable"/>
  <interface name="s" role="client" signature="Service"/>
 </definition>

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

Component references (3)

```
<definition name="SharedHelloWorld">
  <interface name="r" role="server" signature="java.lang.Runnable"/>
  <component name="a" definition="HelloWorld"/>
  <component name="b" definition="HelloWorld">
    <component name="server" definition="a/server"/>
  </component>
  <binding client="this.r" server="a.r"/>
</definition>
```

Arguments

```
<definition name="GenericServerImpl" arguments="impl,header,count">
  <interface name="s" role="server" signature="Service"/>
  <content class="${impl}"/>
  <attributes signature="ServiceAttributes">
    <attribute name="header" value="${header}"/>
    <attribute name="count" value="${count}"/>
  </attributes>
  <controller desc="primitive"/>
 </definition>
```

Usage

```
Map context = new HashMap();
context.put("bootstrap", …); // optional

context.put("impl", "ServerImpl"); // if definition with arguments
context.put("header", "->");
context.put("count", "1");

// four backends can be used (Fractal/Java, static/dynamic)
Factory f =
    FactoryFactory.getFactory(FactoryFactory.FRACTAL_BACKEND);

Component c = (Component)f.newComponent("ClientImpl", context);
```