Monet Execution Manager

Manfred N. Riem

March 17, 2004

What does the Execution Manager (EM) do?

- Execute written or generated plans.
- Call (mathematical) web services as specified in a plan.
- Gives an explanation of an execution.
- Delivers workflow management for plans.
- Delivers a simple calculation environment.

Looking at EM as a black box

As every component in the Monet architecture is implemented as a webservice you can look at it as a blackbox delivering a specific service.

In the case ofl(a)-5(at)-303(oure)-303uredel-303ael-303pla084Td[writte084Td[084

An example of MPL

```
<plan>
  ... parameters ...
  ... variables  ...
  <sequence>
    ... assignments ...
    <sequence>
      <assign>
        <varref name="g"/>
        <invoke url="http://localhost:8080/gcdws/gcdws?MSDL">
          <mapping>
            <map inputname="a"><paramref name="firstValue"/></map>
            <map inputname="b"><paramref name="secondValue"/></map>
           </mapping>
        </invoke>
      </assign>
      <return><varref name="g"/></return>
    </sequence>
  </sequence>
</plan>
```

An example of MEL

```
<monet:query-response ....>
 <monet:executionResponse>
  <monet:problem>
   ...
  </monet:problem>
  <monet:explanation>
   <monet:executionResponse>
    <monet:errorcode>0</monet:errorcode>
    <monet:explanationFormat isProvided="false"/>
    <monet:result>
     <monet:resultFormat type="xsd:int">123</monet:resultFormat>
    </monet:result>
   </monet:executionResponse>
  </monet:explanation>
  <monet:result>
   <monet:resultFormat type="xsd:int">234</monet:resultFormat>
  </monet:result>
 </monet:executionResponse>
</monet:query-response>
```

Example: GCD using an REM webservice (2)

```
  <j:while test="${b ne 0}">
    <i:invoke var="r"
        service="Remws"
        port="RemWebServiceIFPort"
        namespace="http://echoservice.org/wsdl"
        endpoint="http://localhost:8080/remws/remws"
        operation="rem"
        returnType="xsd:int">
      <int_1 type="xsd:int"><j:expr value="${a}"/></int_1>
      <int_2 type="xsd:int"><j:expr value="${b}"/></int_2>
    </i:invoke>
    <j:expr value="r=${r},"/>
    <j:set var="a"><j:expr value="${b}"/></j:set>
    <j:set var="b"><j:expr value="${r}"/></j:set>
    <j:expr value="a=${a},"/>
    <j:expr value="b=${b},"/>
  </j:while>
  <j:expr value="a=${a}"/>
</j:jelly>
```

MPL invocation

MPL defines the invoke element so that a plan can call external webservices. To make it work inside a Jelly context the invocation below needs to be translated into a Jelly invocation.

The translation needs the following things to work:

- The MPL itself.
- The MSDL of any mathematical webservice that needs to be called.
- The WSDL of any mathematical webservice that needs to be called.

Note: the current version of the Execution Manager enables you to call Math-