Workflow Management System

The Workflow Management System (WfMS) project is developed within the CoreGRID european project. The main goal of this project is to provide an effective solution to run *complex scientific workflows* modeled with *Petri Nets* taking full advantage of the *distributed* and *etherogeneous* nature of the Grid. One of the design principle is the *neutrality* towards the underlying mechanism for task execution, in order not to compromise *interoperability* with multiple infrastuctures.

The project also aims at *language interoperability*, placing attention to workflow description languages and introducing *language translators*. The internal representation of a workflow is based on the High Level Petri Nets (HLPN) formalism due to its formal semantics and the existence of several analysis tools. The reference language of the WfMS is the Grid Workflow Description Language (GWorkflowDL) which is based on the HLPN formalism.

The fist prototype of the system have been tested with workflows accessing resources available on the Grid provided by the EGEE project, a large and relatively mature infrastucture. In particular, the execution of Grid jobs is performed by relying on the gLite Workload Management System (WMS) through its Web Service interface (WMProxy).

Contacts

INFN Cnaf Bologna
Simone Pellegrini, Francesco Giacomini

Main Web Site

http://wfms.forge.cnaf.infn.it

Papers

CGW07
Simone Pellegrini, Francesco Giacomini, Antonia Ghiselli: *A Practical Approach for a Workflow Management System*. In Proceedings of the CoreGRID Workshop 2007, Dresden, 2007.

CGIW08
Simone Pellegrini, Andreas Hoheisel, Francesco Giacomini, Antonia Ghiselli: *Using GWorkflowDL for Middleware-Independent Modeling and Enactment of Workflows*. Submitted for the CoreGRID Integration Workshop 2008, Crete, 2008.

WaGe08 `[coming]`
Simone Pellegrini, Francesco Giacomini: *Desing of a Petri Net-based Workflow Engine*. Submitted for the The 3rd International Conference on Grid and Pervasive Computing (GPC 2008), Kunming (China), 2008.

CG08 `[under reviewing process]`
Simone Pellegrini, Francesco Giacomini: *GWorkflowDL: A Multi-purpose Language for Scientific Workflow Enactment*. Submitted for the 3rd CoreGRID Workshop on Grid Middleware, Barcellona, 2008.

Concepts

Petri Nets in workflow modeling

Recent studies have demostrated that the modling capabilities of Petri Nets outperforms other formalisms tanks to the following properties:

1. the formal semantics despite the graphical nature,

2. state-based structure (as opposed to the event-based one),
3. the availability of many analysis techniques.

A small tutorial about the Petri Nets, and their usage in the workflow management can be found here.

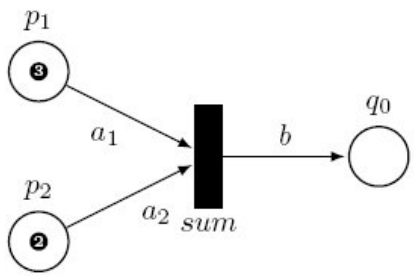
More suitable than Petri Nets, the High Level Petri Nets (HLPN) formalism can be used for workflow modeling. The HLPN term is used for many Petri Nets formalisms that extend the basic P/T net formalism. This includes *coloured* Petri Nets, *hierarchical* Petri Nets, and *timed* Petri Nets. The Grid Workflow Description Language is based on HLPN.

The Grid Workflow Descpription Language

The GWorkflowDL consists of two parts:

1. a generic part, used to define the structure of the workflow, reflecting the data and control flow in the application,
2. a middleware-specific part (extensions) that defines how the workflow should be executed in the context of a specific Grid computing middleware.

Considering the generic Petri Net depicted in the following figure:



The generic part of the workflow can be represented in the GWorkflowDL language as:

```
<workflow>
   <place ID="p1">
     <token><data><t1 xsd:type="xs:int">3</t1></data></token>
   </place>
   <place ID="p2">
     <token><data><t2 xsd:type="xs:int">2</t2></data></token>
   </place>
   <place ID="q0" />
   <transition ID="sum">
     <inputPlace placeID="p1" edgeExpression="a1"/>
     <inputPlace placeID="p2" edgeExpression="a2"/>
     <outputPlace placeID="q0" edgeExpression="b"/>
     <operation /> <!-- generic operation -->
   </transition>
</workflow>
```

In the workflow there are no information about the operation associated to the Petri Net transiion *T*. More detailed information are provided by the *concrete* part of the description. A concrete operation could be the invocation of a *Web Service*, or the *remote execution* of a program or the invocation of a *local routine*.

The enactment process, performed by the WfMS, is responsable of mapping abstract operations -- associated to a Petri Net transisions -- to concrete operations. For example the *plus* operation represented in the above workflow can be mapped to a Web Service invocation as described in the following concrete workflow:

```
<workflow>
```

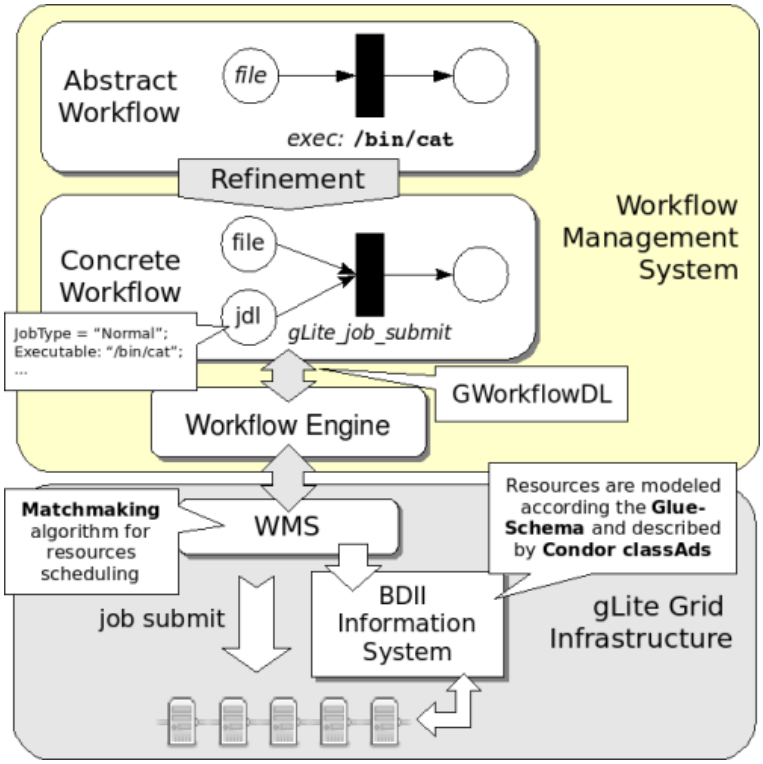
```
   <place ID="p1">
     <token><data><t1 xsd:type="xs:int">3</t1></data></token>
   </place>
   <place ID="p2">
     <token><data><t2 xsd:type="xs:int">2</t2></data></token>
   </place>
   <place ID="q0" />
   <transition ID="sum">
      <inputPlace placeID="p1" edgeExpression="a1"/>
      <inputPlace placeID="p2" edgeExpression="a2"/>
      <outputPlace placeID="q0" edgeExpression="b"/>
      <operation>
         <wsOperation wsdl="http://localhost/plus?wsdl" operationName="plus">
            <in>n1</in>
            <in>n2</in>
            <out>q1</out>
         </wsOperation>
      </operation>
   </transition>
</workflow>
```

Or, for example, mapped to the local method invocation as depicted in the following concrete workflow:

```
<workflow>
   <place ID="p1">
     <token><data><t1 xsd:type="xs:int">3</t1></data></token>
   </place>
   <place ID="p2">
     <token><data><t2 xsd:type="xs:int">2</t2></data></token>
   </place>
   <place ID="q0" />
   <transition ID="sum">
      <inputPlace placeID="p1" edgeExpression="a1"/>
      <inputPlace placeID="p2" edgeExpression="a2"/>
      <outputPlace placeID="q0" edgeExpression="b"/>
      <operation>
         <pyOperation operation="b = a1 + a2" />
      </operation>
   </transition>
</workflow>
```

The WfMS and the EGEE/gLite Grid Middleware

The first prototype of the WfMS has been tested with workflows accessing the resources provided by the EGEE/gLite middleware. The gLite middleware takes care of finding the best available resources considering a set of users requirements and preferences (such as CPU architecture, OS, current load) disburdening the WfMS from the resources management. In the following picture an high-level view of the system is provided:



The core component of the system is the workflow engine which *execute* concrete workflows written in GWorkflowDL. The engine simply schedules the activities accoriding to Petri Nets model and demand their execution to the WMS. However, the engine, by design, has no knowledge about the underying middleware and Grid operations, such as job submission, monitoring, etc... muse be expressed in terms of atomic operations the engine can execute. For example the submission of a job to the gLite middlware requires several web services to be invoked in sequence; sequence which can be easily represented using a workflow. As a conseguence, providing to the engine few *atomic* functionalities, such as web service interaction and local method execution make it possible to execute complex scientific processes simply by composing atomic operations as depicted below.

Consider the following abstract workflow, it consists in the remote execution of the command: **/bin/cat data.dat**.

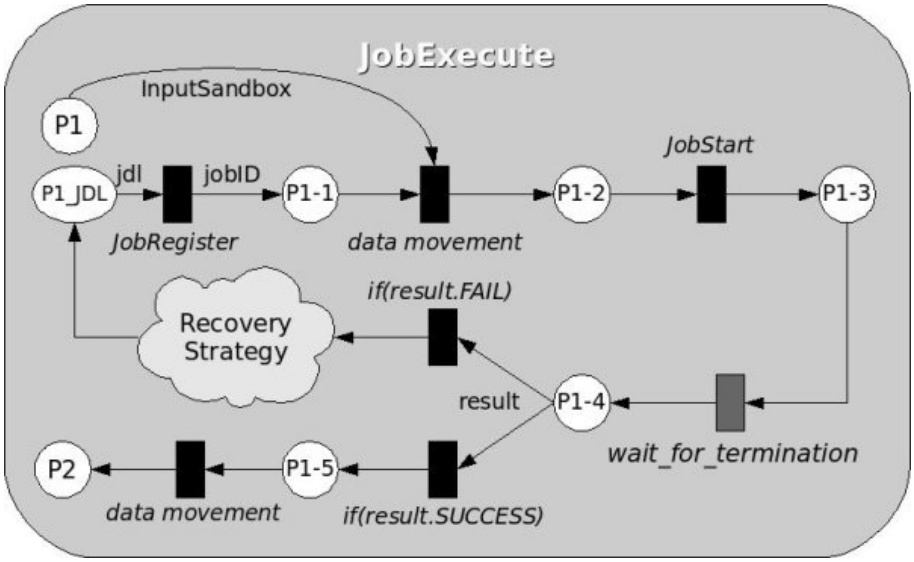
```
<workflow>
   <place ID="p1">
     <token><data><file xsd:type="xsd:string">file://home/data.dat</file></data></token>
   </place>
   <place ID="p2" />
   <transition ID="cat">
      <inputPlace placeID="p1" edgeExpression="in"/>
      <outputPlace placeID="p2" edgeExpression="out"/>
      <operation/>
   </transition>
</workflow>
```

If we are going to address the gLite middleware the workflow can be converted into the following concrete representation:

```
<workflow>
   <place ID="p1">
     <token><data><file xsd:type="xsd:string">file://home/data.dat</file></data></token>
   </place>
   <place ID="p1_jdl">
     <token><data>
        <jdl xsd:type="xsd:string">
           Type = "Job";
```

```
            JobType = "Normal";
            Executable = "/bin/cat";
            Arguments = "data.dat";
            StdOutput = "dump.txt";
            InputSandbox = {"data.dat"};
            OutputSandbox = {"dump.txt"};
            Rank = -other.GlueCEStateEstimatedResponseTime;
            Requirements = (other.GlueCEInfoHostName == "ce06-lcg.cr.cnaf.infn.it");
         </jdl></data></token>
   </place>
   <place ID="p2" />
   <transition ID="cat">
      <inputPlace placeID="p1" edgeExpression="in"/>
      <inputPlace placeID="p1_jdl" edgeExpression="jdl"/>
      <outputPlace placeID="q0" edgeExpression="out"/>
      <operation>
         <swOperation name="JobExecute" />
      </operation>
   </transition>
</workflow>
```

The `swOperation` allows to invoke sub-workflows. A sub-workflow implements a particular operation on a specific infrastucure. In this case, the `JobExecute` workflow describes (using the Petri Nets formalism) the operation to perform in order to execute a task in the gLite Grid middleware. The sequence of operation is depicted in the following figure:



The `JobRegister`, `JobSubmit` operations are provided by the WMS via its Web Service interface, this make it possible to express the `JobExecute` sub-workflow in terms of atomic operations (actually Web Service invocation) the engine can understand and execute. The execution of a job in the gLite middleware consists in a `jobSubmit` (which semantics can be also obtained by invoking -- in sequence -- `jobRegister` and a `jobStart`) and the monitor of the job activity until done. Job monitoring can be done via the WMS using the Logging and Bookkeeping Service (LB) also available via a Web Service interface. The `wait_for_termination` task can be modelled using a sub-workflow which implement a specific monitoring strategy (*polling* or *notification-based*).

Workflow Engine

The engine, represent the core component of the WfMS. It is based on the HLPN formalism and has the responsability to schedule workflow tasks accoring to their interdependencies keeping the overall state of the workflow execution. The engine deals with the non-determinism in Petri Nets providing an unique design capable of guarantee state consistency and ideally inifinite parallelism in task execution.

This topic: EgeeJra1It > WorkflowManagementSystem
Topic revision: r7 - 2008-04-15 - SimonePellegrini