

gLite Workload Management System

Emidio Giorgio
INFN Catania

www.eu-egee.org

Workload Management System

WMS Architecture

Job state machine

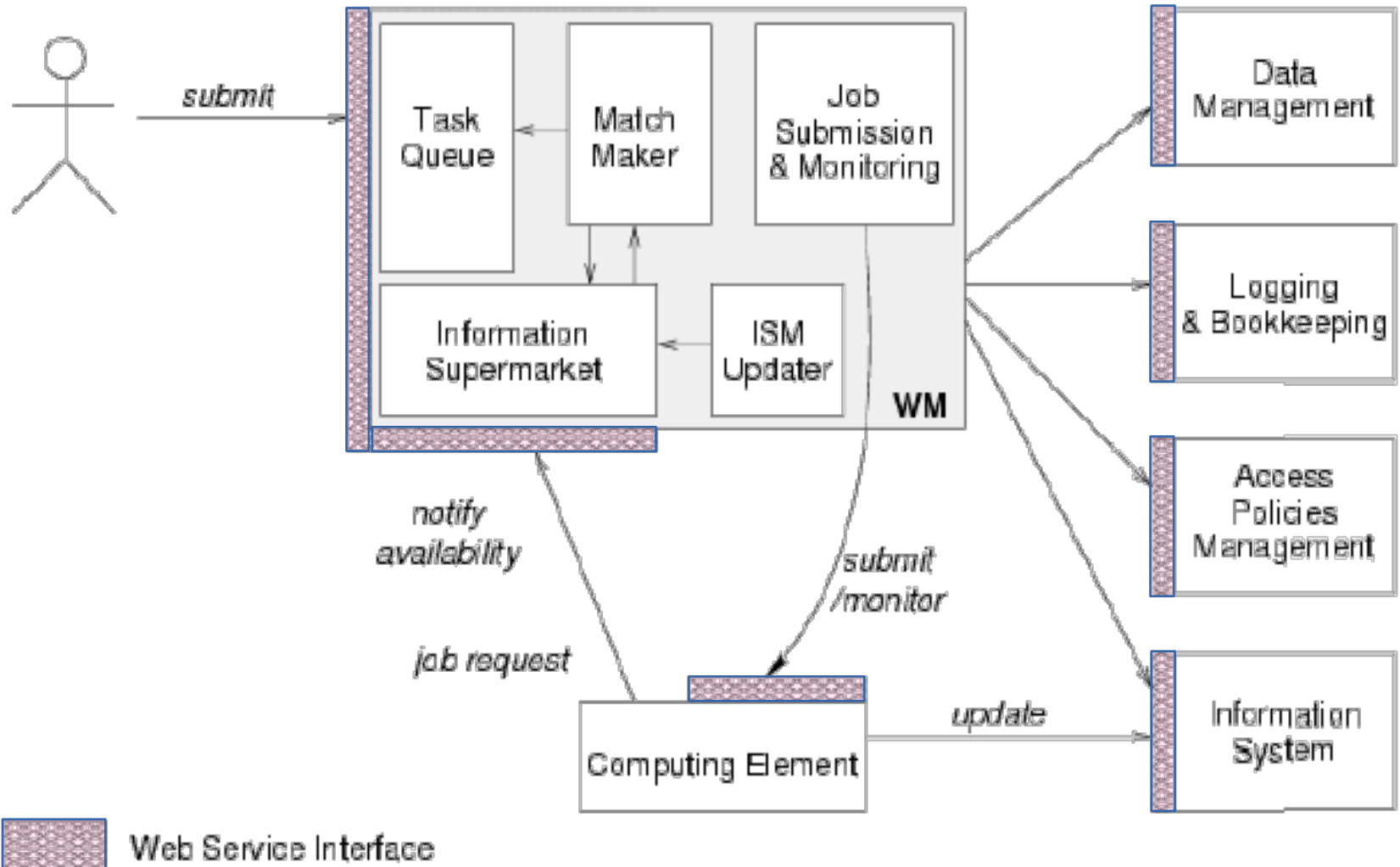
Job Description Language Overview

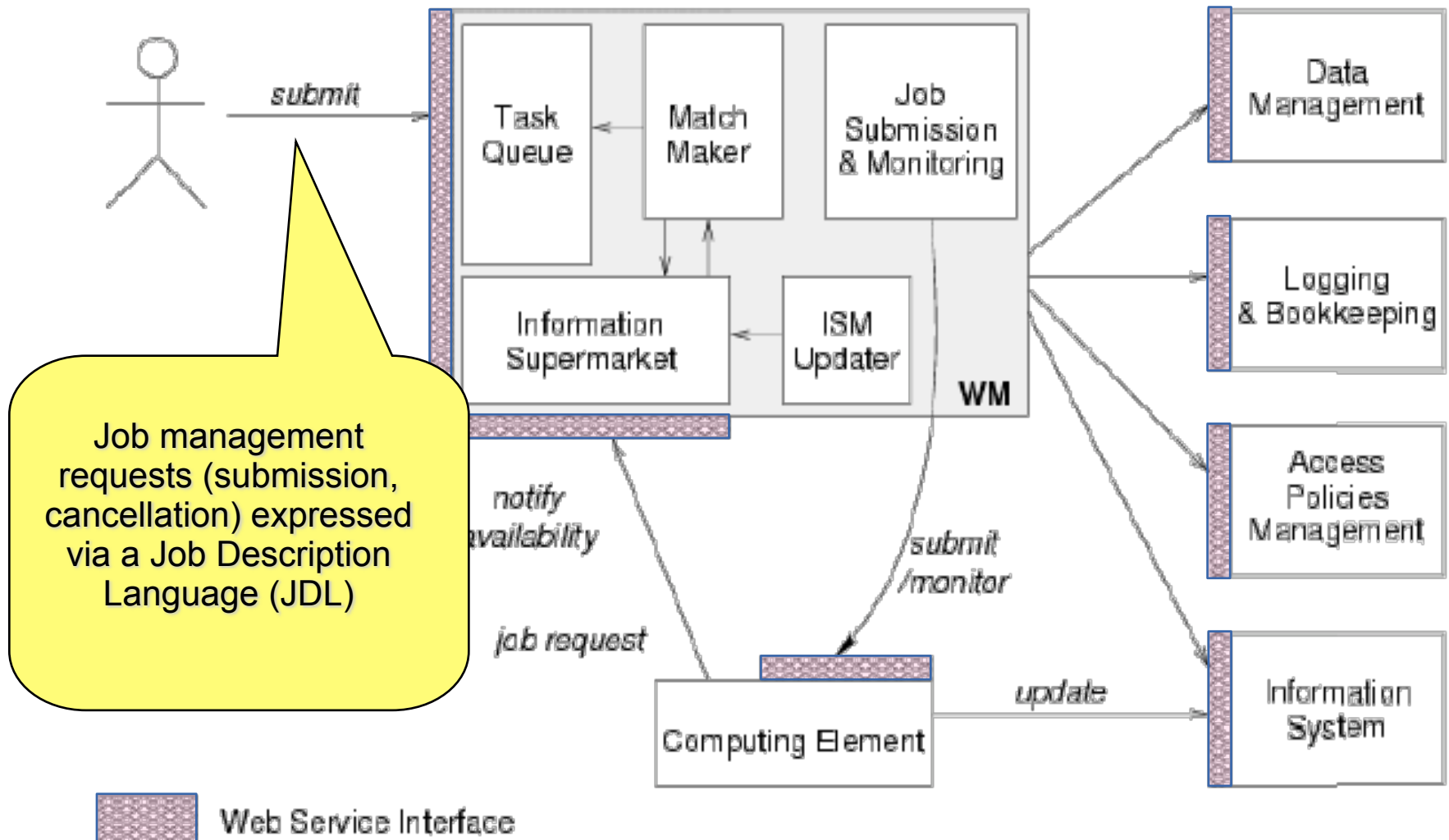
The **Workload Management System** (WMS) comprises a set of Grid middleware components responsible for distribution and management of tasks across Grid resources.

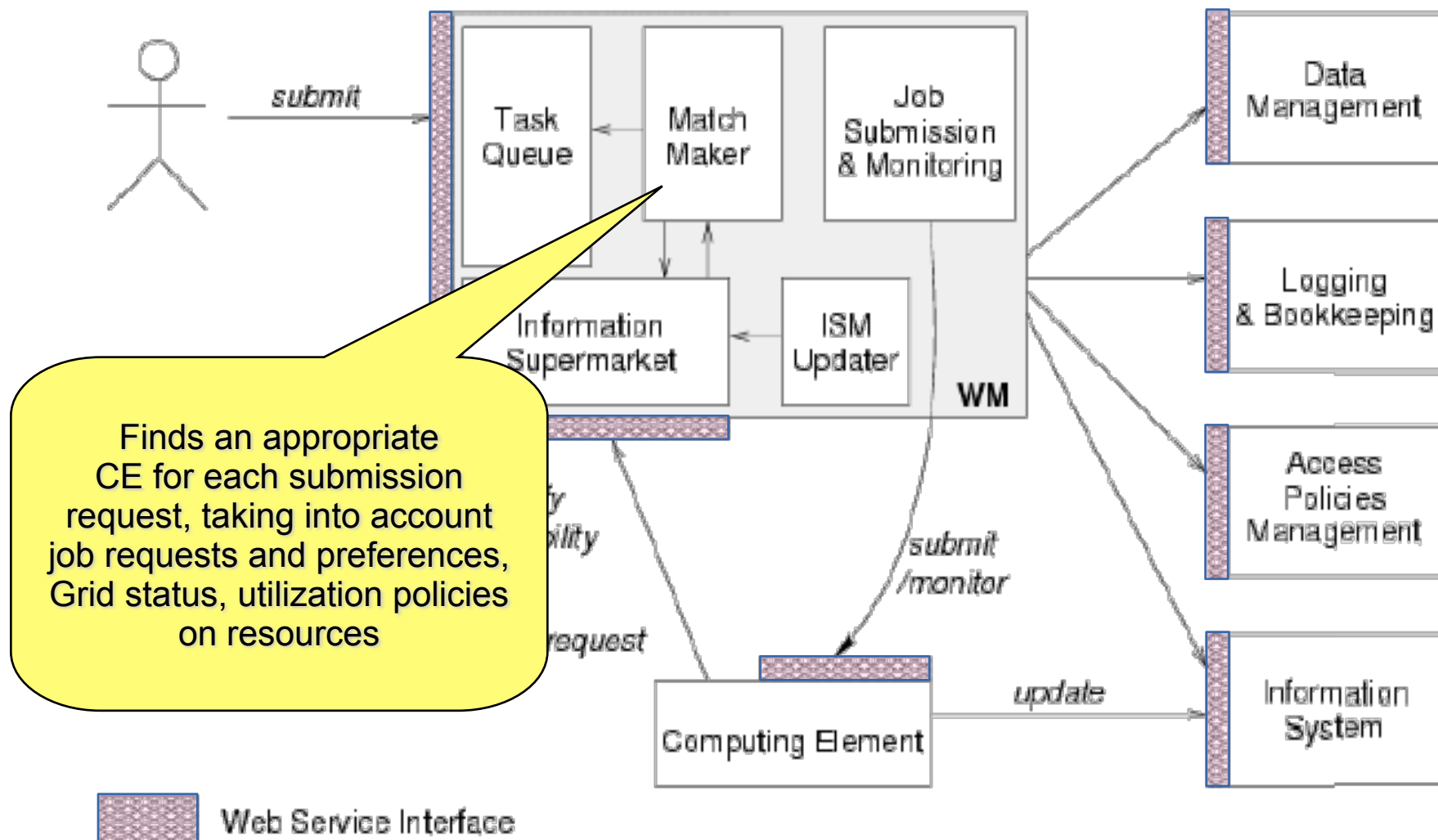
The purpose of the Workload Manager (WM) is to accept and satisfy requests for job management coming from its clients meaning of the submission request is to pass the responsibility of the job to the WM.

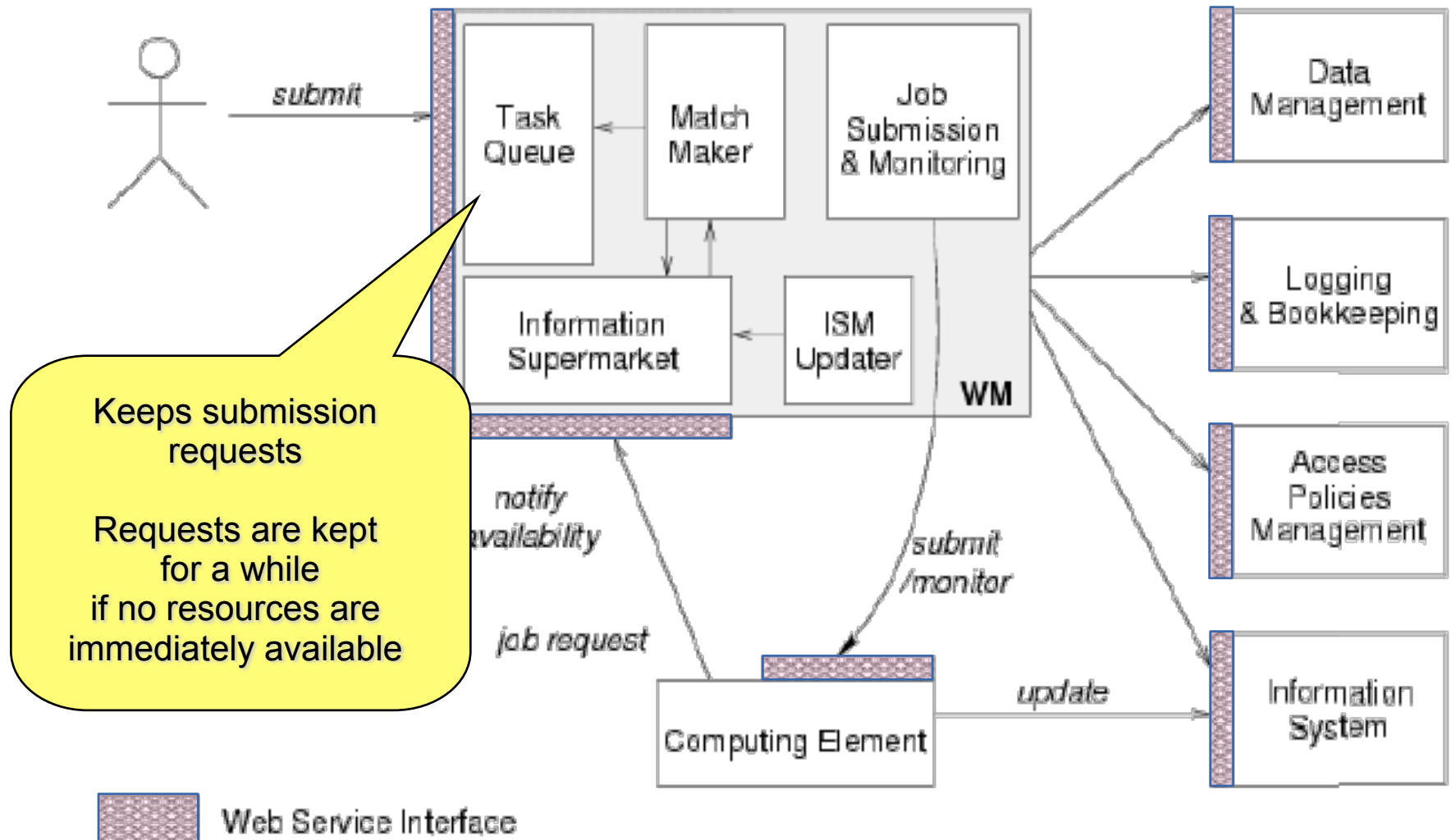
WM will pass the job to an appropriate CE for execution
taking into account requirements and the preferences expressed in the job description file

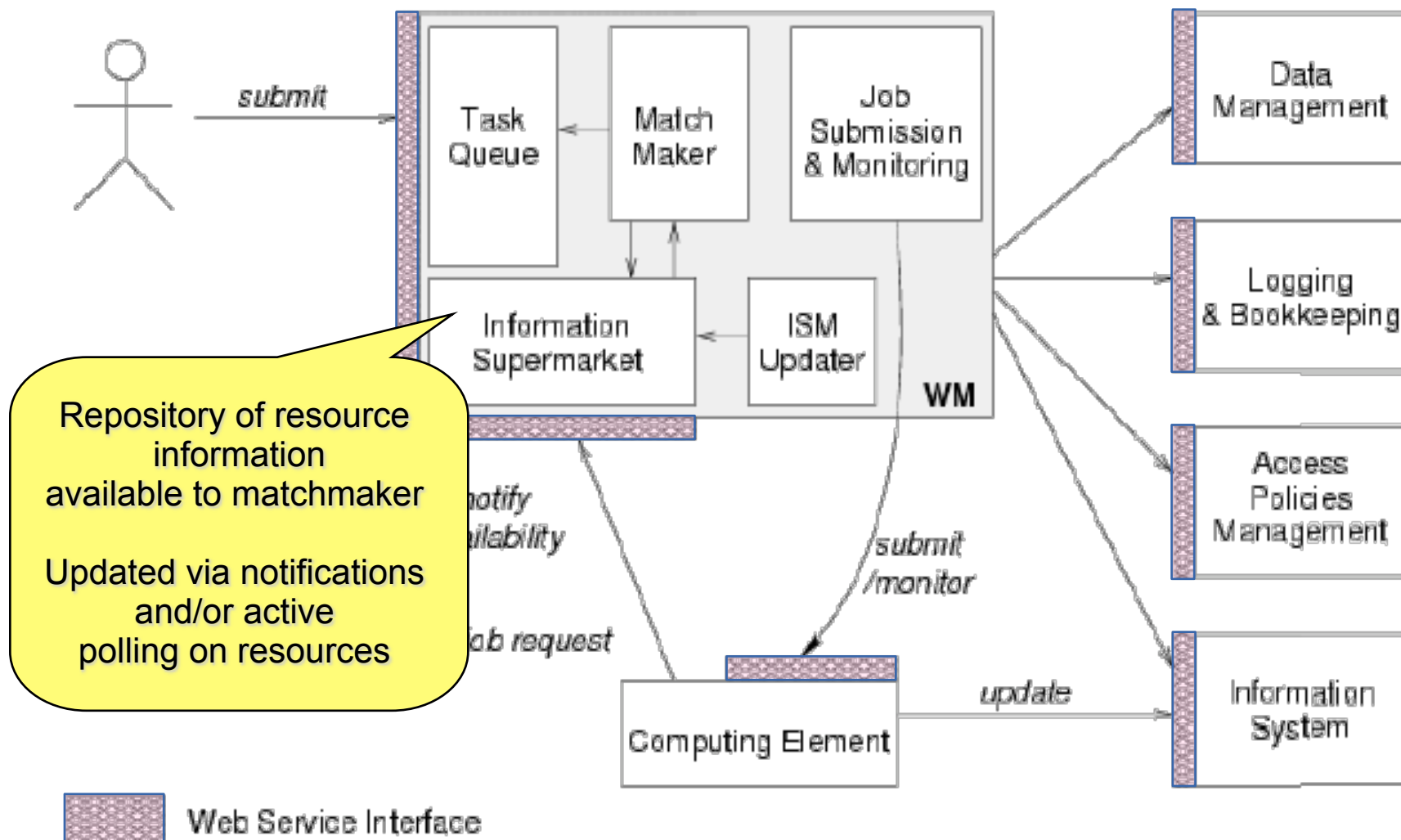
The decision of which resource should be used is the outcome of a **matchmaking** process.

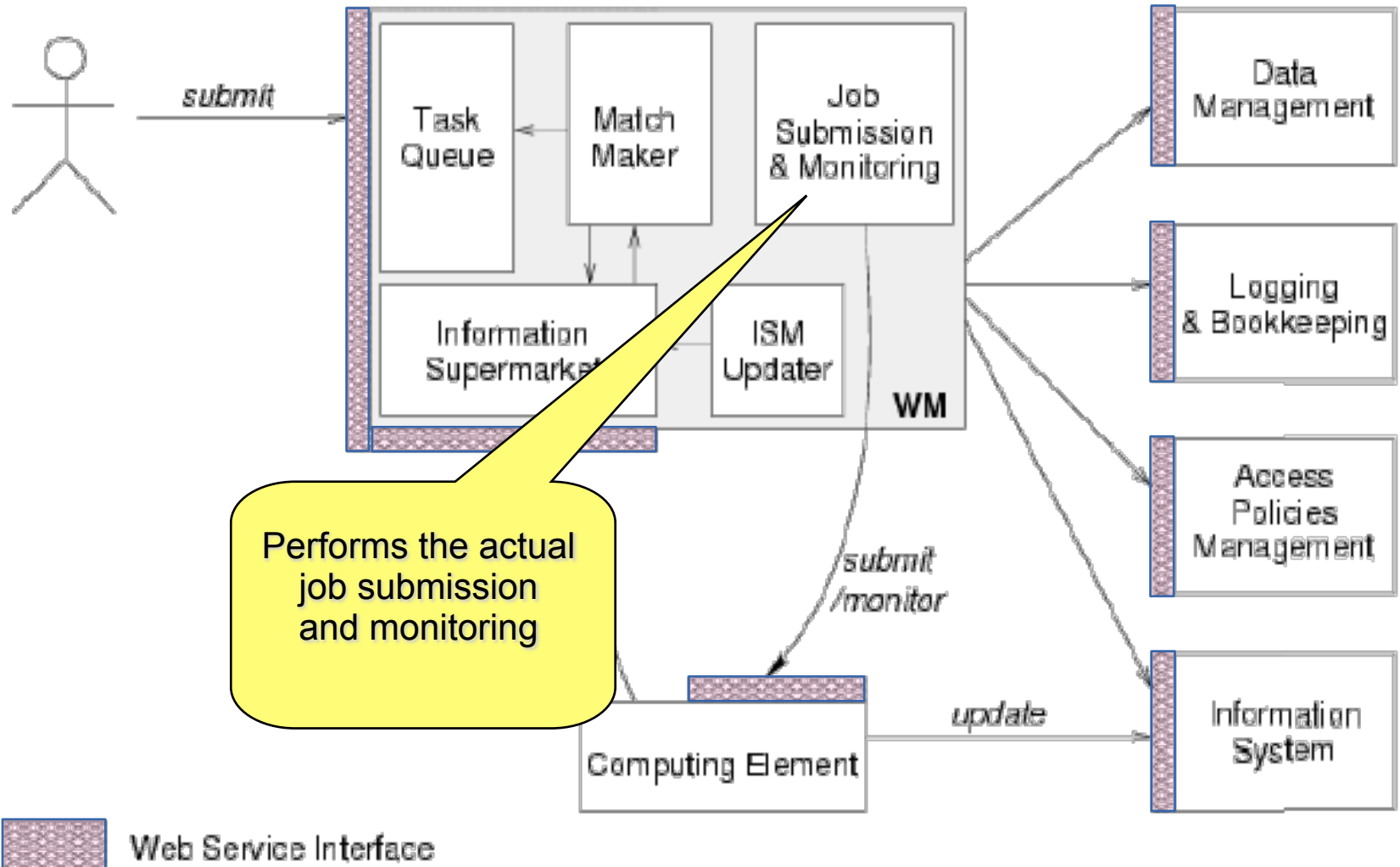


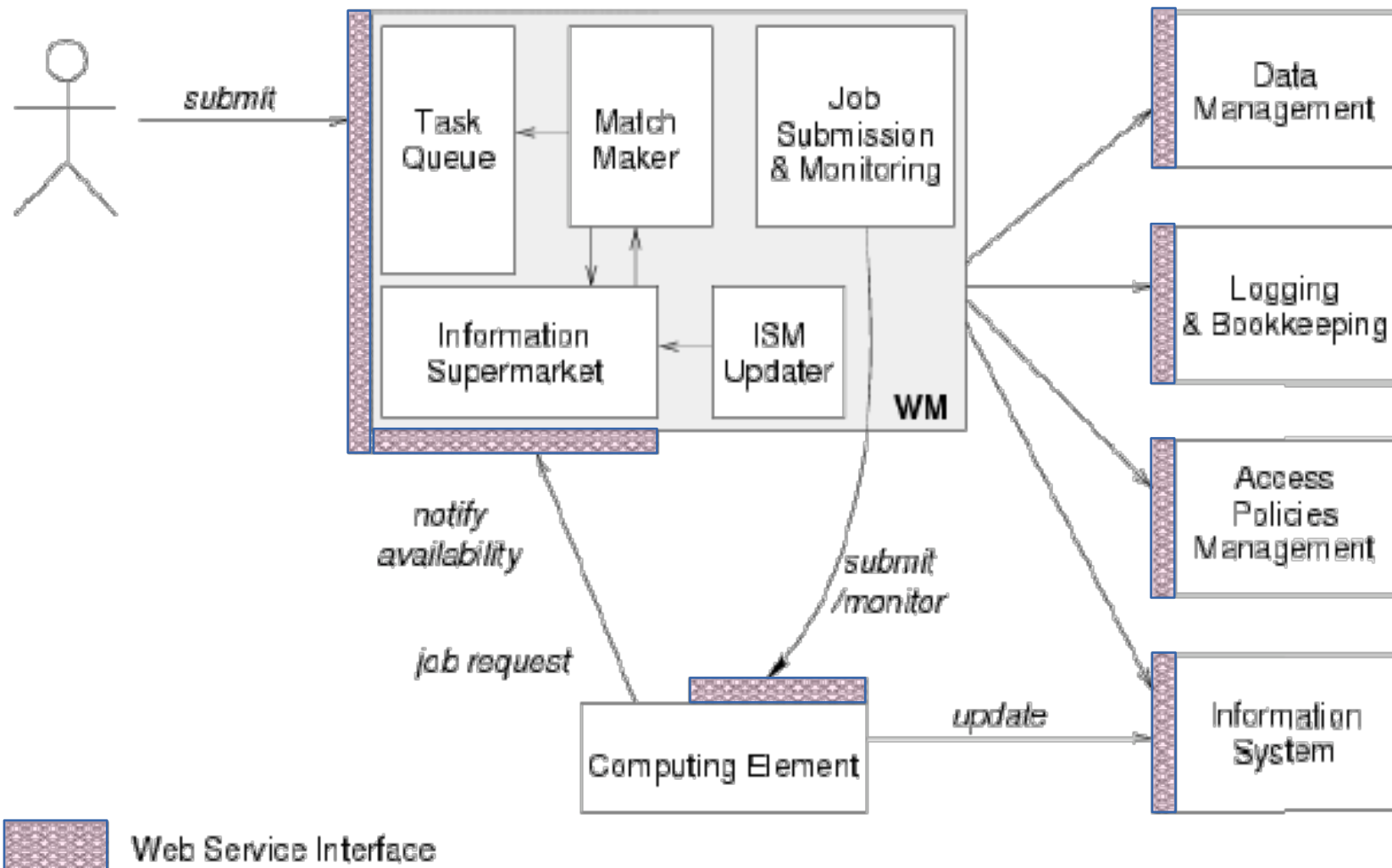












Job Description Language

In gLite, **Job Description Language (JDL)** is used to describe jobs for execution on Grid.

The JDL adopted within the gLite middleware is based upon Condor's **CLASSified Advertisement language (ClassAd)**.

A ClassAd is a record-like structure composed of a finite number of attributes separated by semi-colon (;)

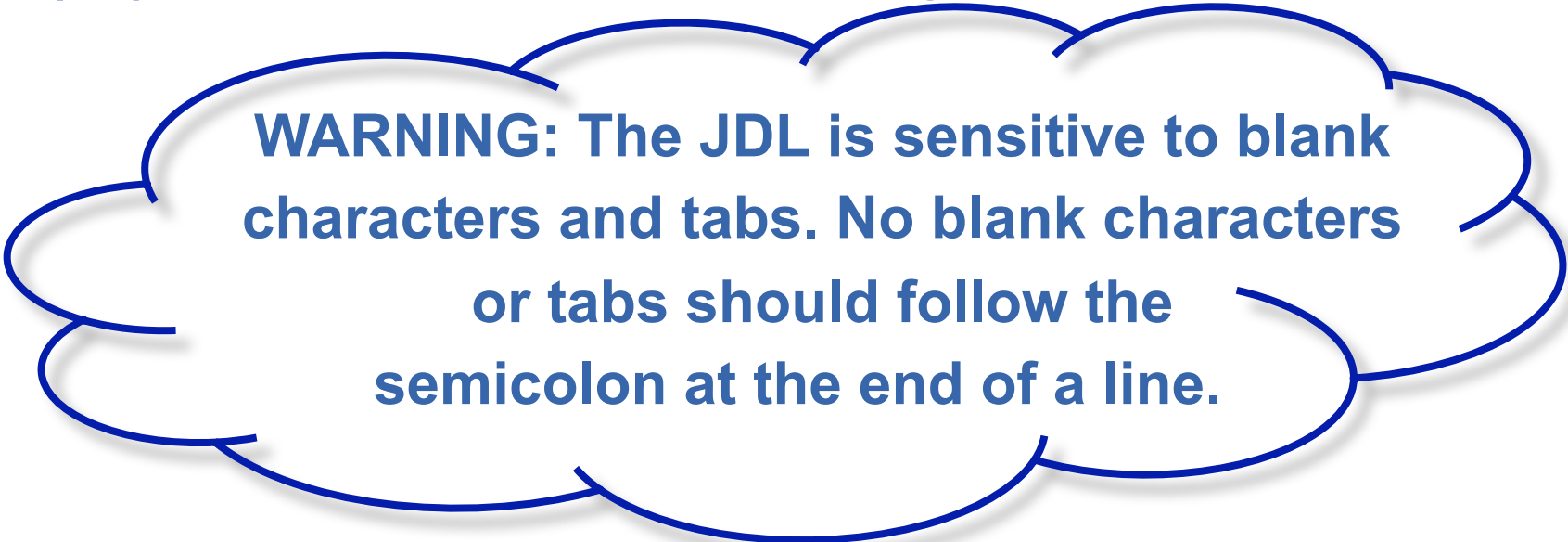
A ClassAd is highly flexible and can be used to represent arbitrary services

*The JDL is used in gLite to specify the job's characteristics and constraints, which are used during the **match-making process** to select the best resources that satisfy job's requirements.*

The **JDL syntax** consists on statements like:

Attribute = value;

Comments must be preceded by a sharp character
(#) or have to follow the C++ syntax



WARNING: The JDL is sensitive to blank characters and tabs. No blank characters or tabs should follow the semicolon at the end of a line.

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

Executable indicates
which file will be
executed remotely

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=.:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

Environment allows to specify env. variables which will be set at run time


```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=.:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

Arguments appends a string (to be used as argument) to **Executable**

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

StdOutput is the
remote file where
output will be
redirected

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

StdError is the remote file where std error will be redirected

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

InputSandbox defines
a set of local files that
you want to be staged
remotely for execution

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=.:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

OutputSandbox
defines a set of remote
files that you want to
get back after
execution

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=.:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

Requirements allows to specify a set of characteristic (hardware or software that you wish for the resource).

```
Type = "Job";
JobType = "Normal";
Executable = "startGen4.sh";
Environment = {"CLASSPATH=./gfal.jar:./
gint.jar","LD_LIBRARY_PATH=:
$LD_LIBRARY_PATH","LCG_GFAL_VO=gilda","LCG_RFIO_TYPE=dpm"};
Arguments = " 0 0 10 4 10000 aliserv6.ct.infn.it lfn:/grid/
gilda/valeria/2000pillar.dat /gilda/issgc07/";
StdOutput = "sample.out";
StdError = "sample.err";
InputSandbox =
{"startGen4.sh","gint.jar","gfal.jar","libGFalFile.so"};
OutputSandbox = {"sample.err","sample.out"};
Requirements =
Member("GLITE-3_0_0",other.GlueHostApplicationSoftwareRunTim
eEnvironment);
```

- If your job needs a file stored somewhere, you can specify its LFN :
- The **file will not be copied** but your job scheduled to a CE near the SE holding that file
- That is crucial when dealing with large files

```
DataRequirements = {  
    [  
        InputData = {"lfn:/grid/gilda/emidio/test.txt"};  
        DataCatalogType = "DLI";  
        DataCatalog = "http://lfc-gilda.ct.infn.it:8085";  
    ]  
};  
DataAccessProtocol = {"rfio","gsiftp"};
```


- **Rank** : allows to override UI's default for fitness function on which resources are classified

```
Rank = ( other.GlueCEStateWaitingJobs == 0 ?
other.GlueCEStateFreeCPUs : -
other.GlueCEStateWaitingJobs );
```

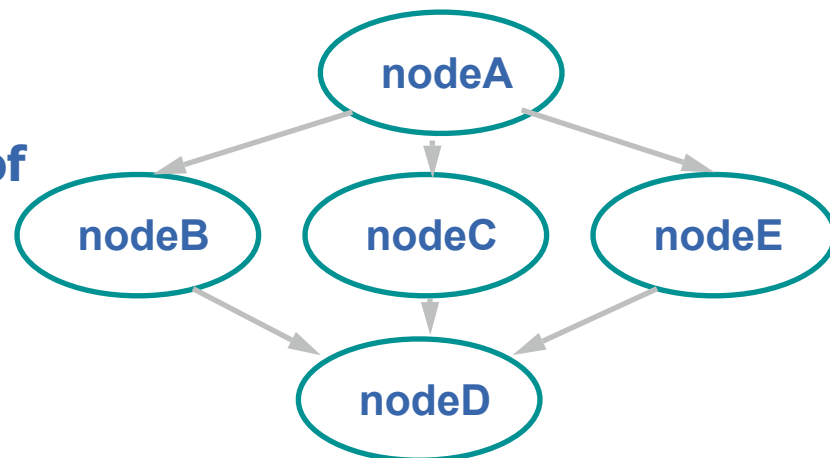
- **RetryCount** : override default for times that a job will be resubmitted after the first failure

```
RetryCount = 7
```

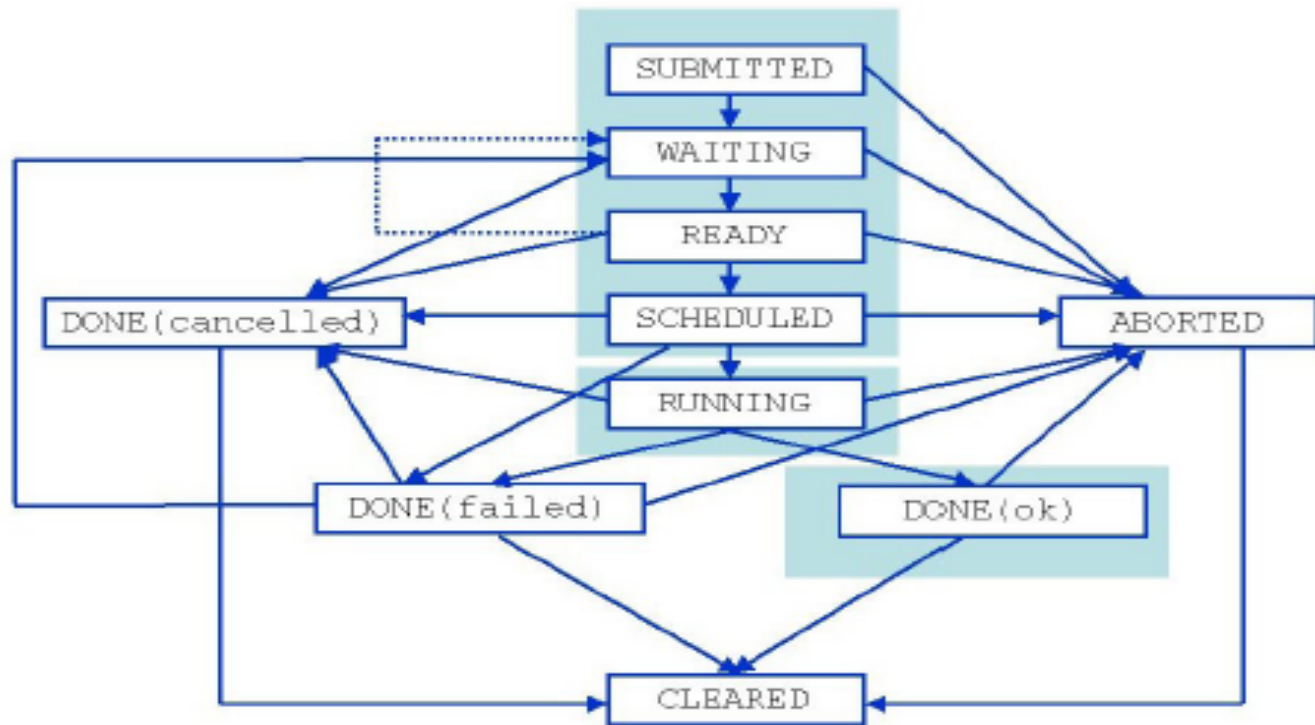
- **Requirements** : a wide set of attributes, as they are published from the BDII, can be required. Regular expressions can be even set, and/or combined with

```
Requirements =
(RegExp( ".*pdc.kth.se", other.GlueCEUniqueID ) );
```

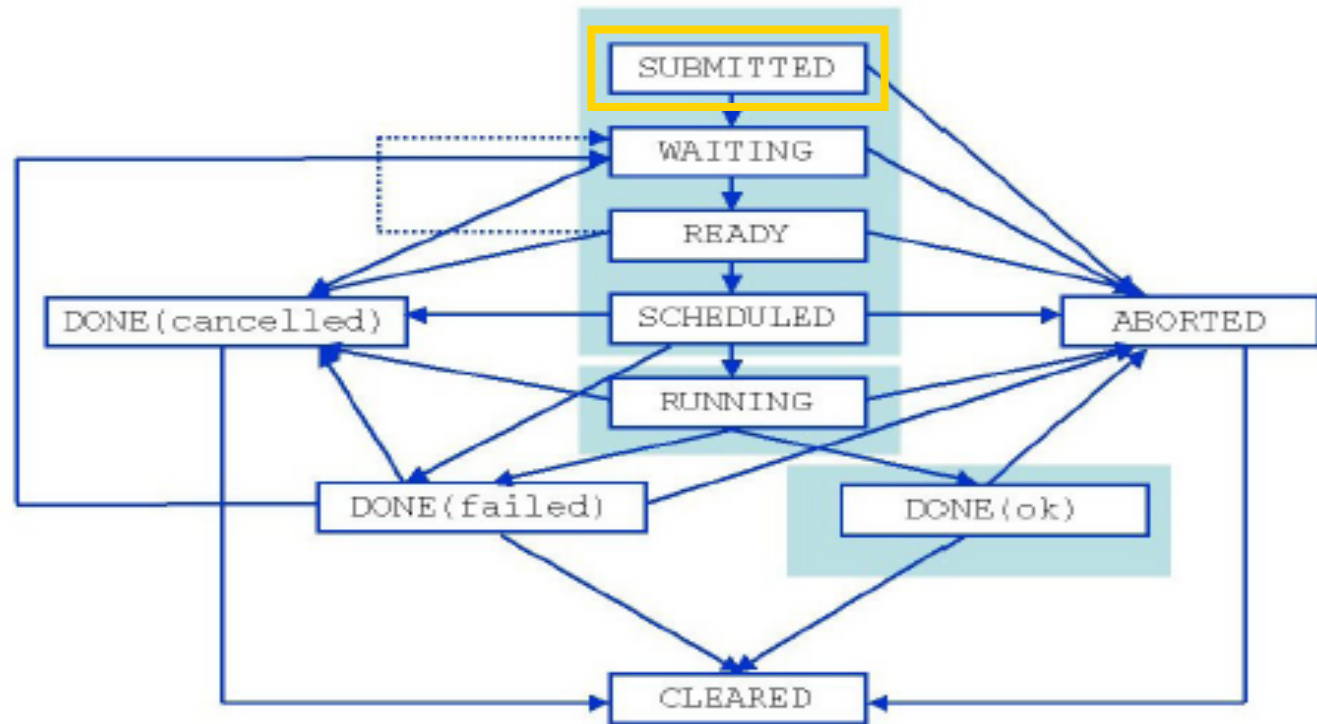
- With a single request, multiple jobs can be generated and executed
- **Direct Acyclic Graph (DAG)** is a set of jobs where the input, output, or execution of one or more jobs depends on one or more other jobs
- A **Collection** is a group of jobs with no dependencies
 - basically a collection of JDL's
- A **Parametric job** is a job having one or more attributes in the JDL that vary their values according to parameters
- Using compound jobs it is possible to have one shot submission of a (possibly very large, up to thousands) group of jobs
 - Submission time reduction
 - Single call to WMPProxy server
 - Single Authentication and Authorization process
 - Sharing of files between jobs
 - Availability of both a single Job Id to manage the group as a whole and an Id for each single job in the group



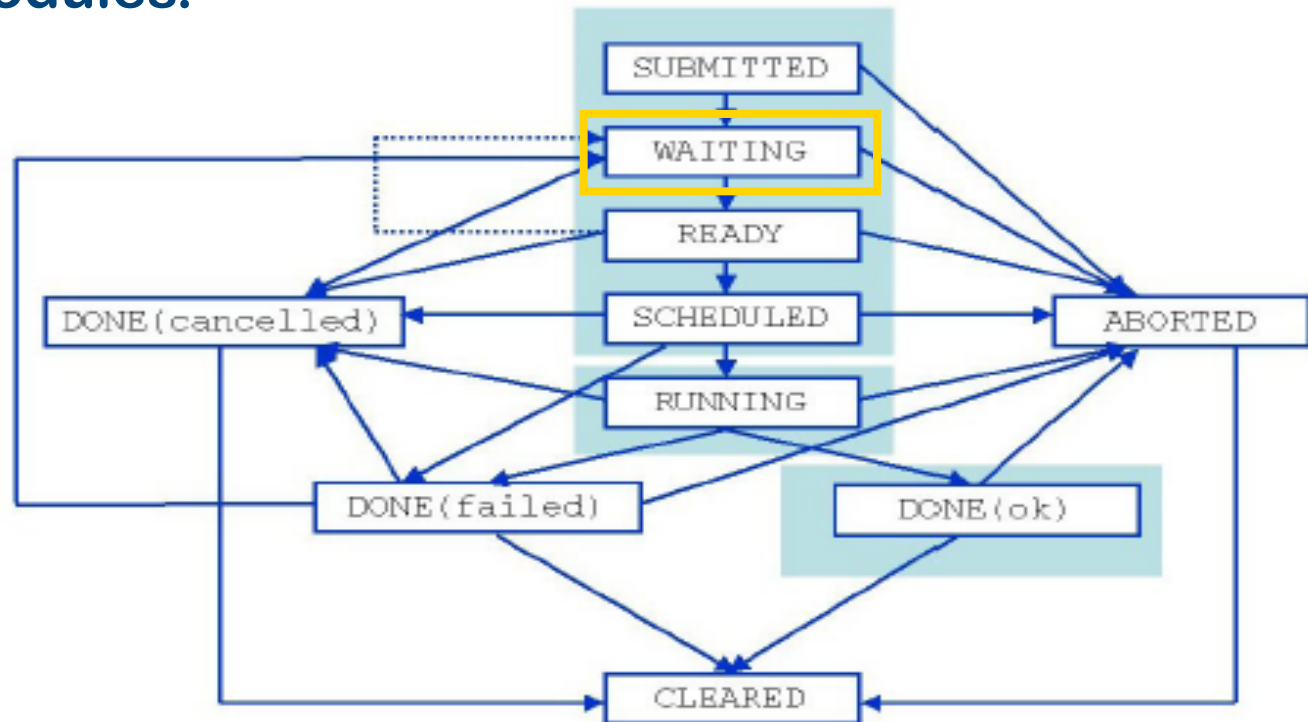
Job State machine



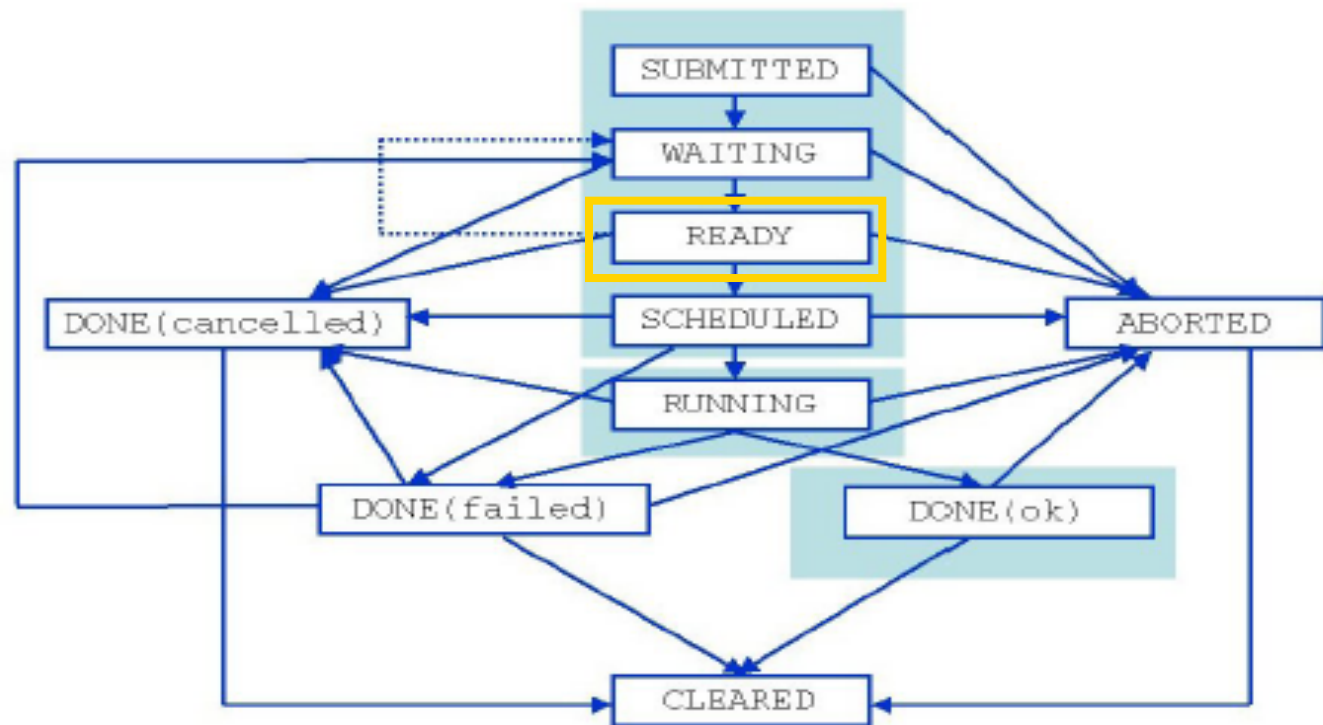
Submitted job is entered by the user to the User Interface but not yet transferred to Network Server for processing



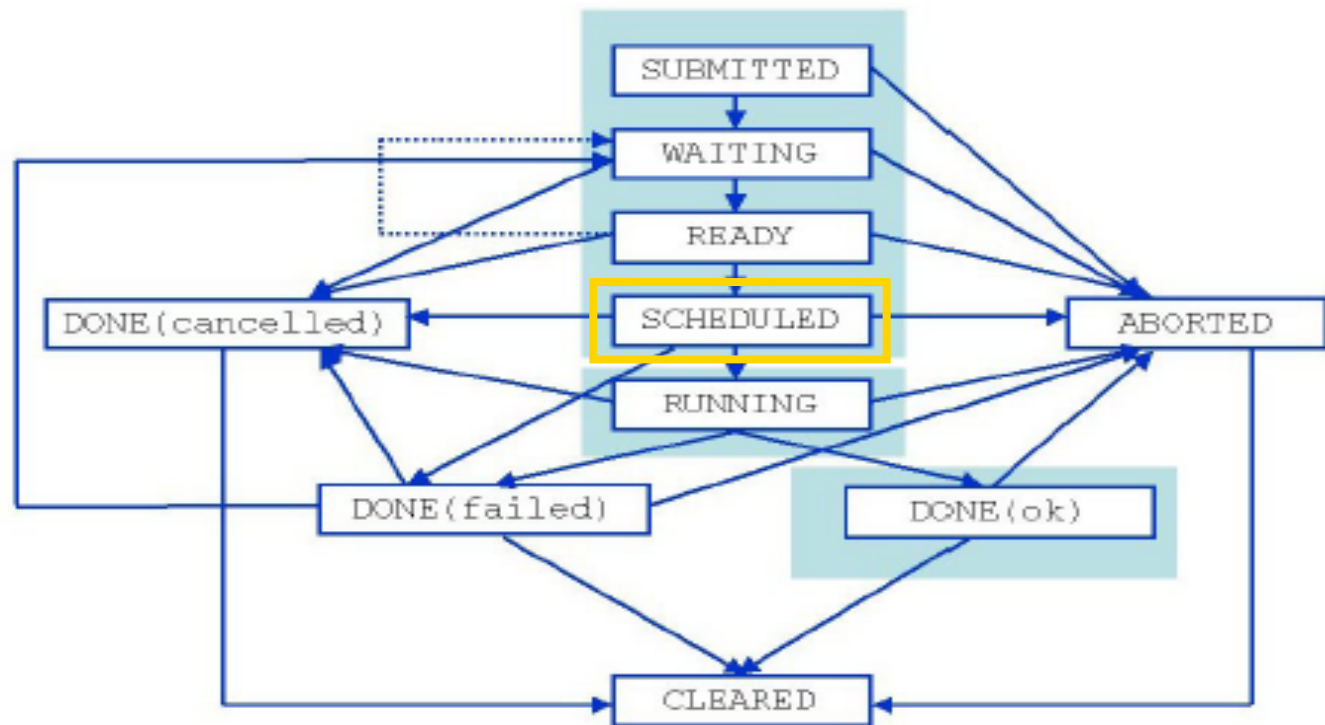
Waiting job accepted by NS and waiting for Workload Manager processing or being processed by WMHelper modules.



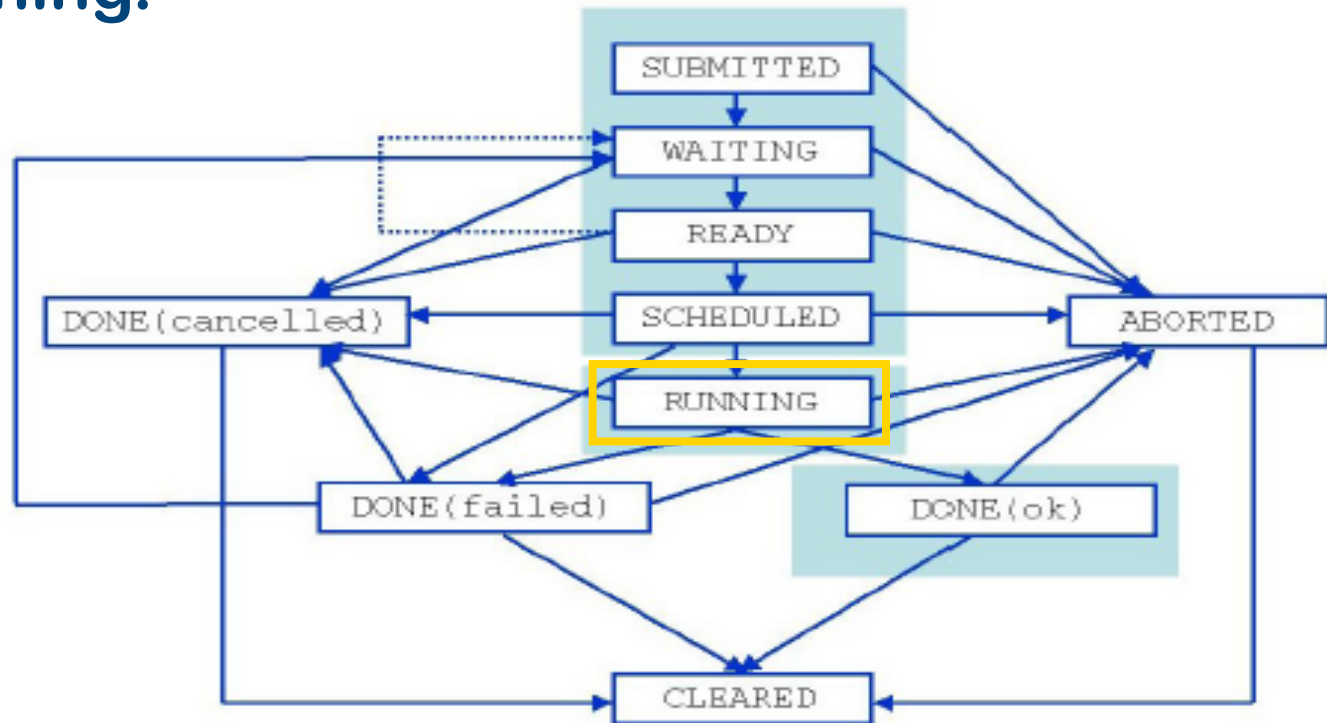
Ready job processed by WM
but not yet transferred to the
CE (local batch system
queue).



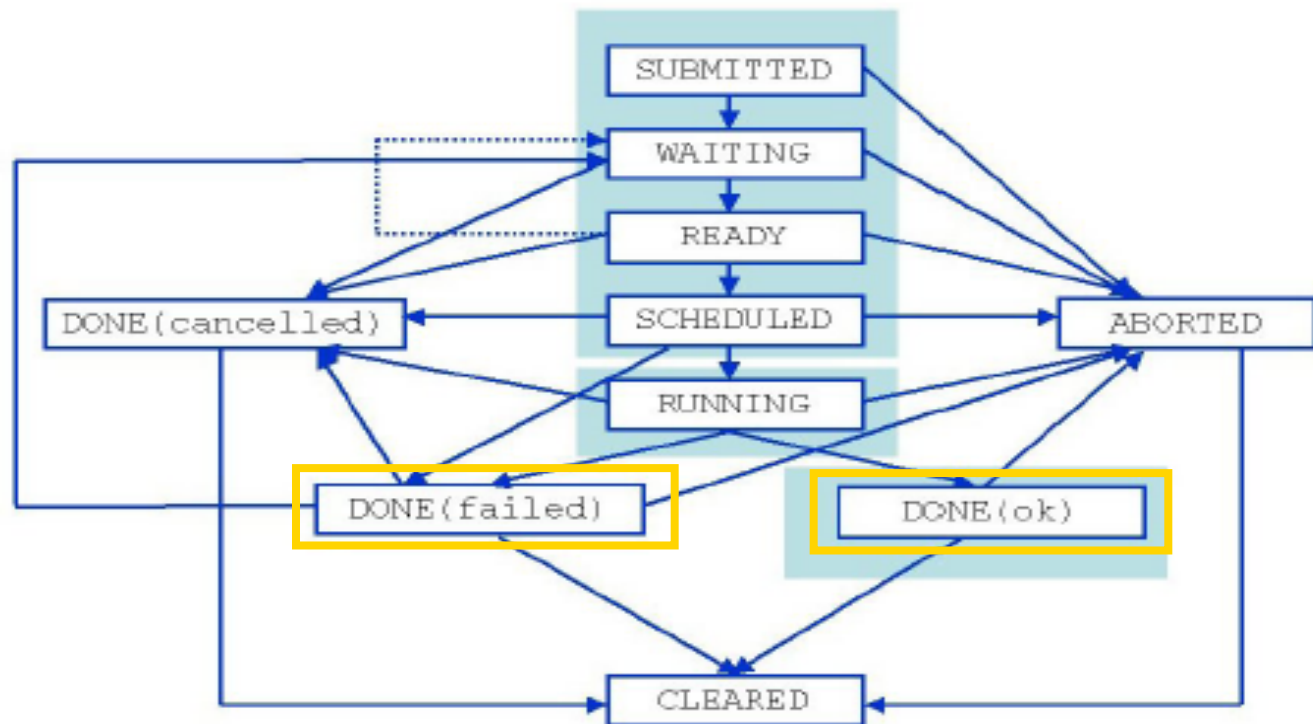
Scheduled job waiting in the queue on the CE.



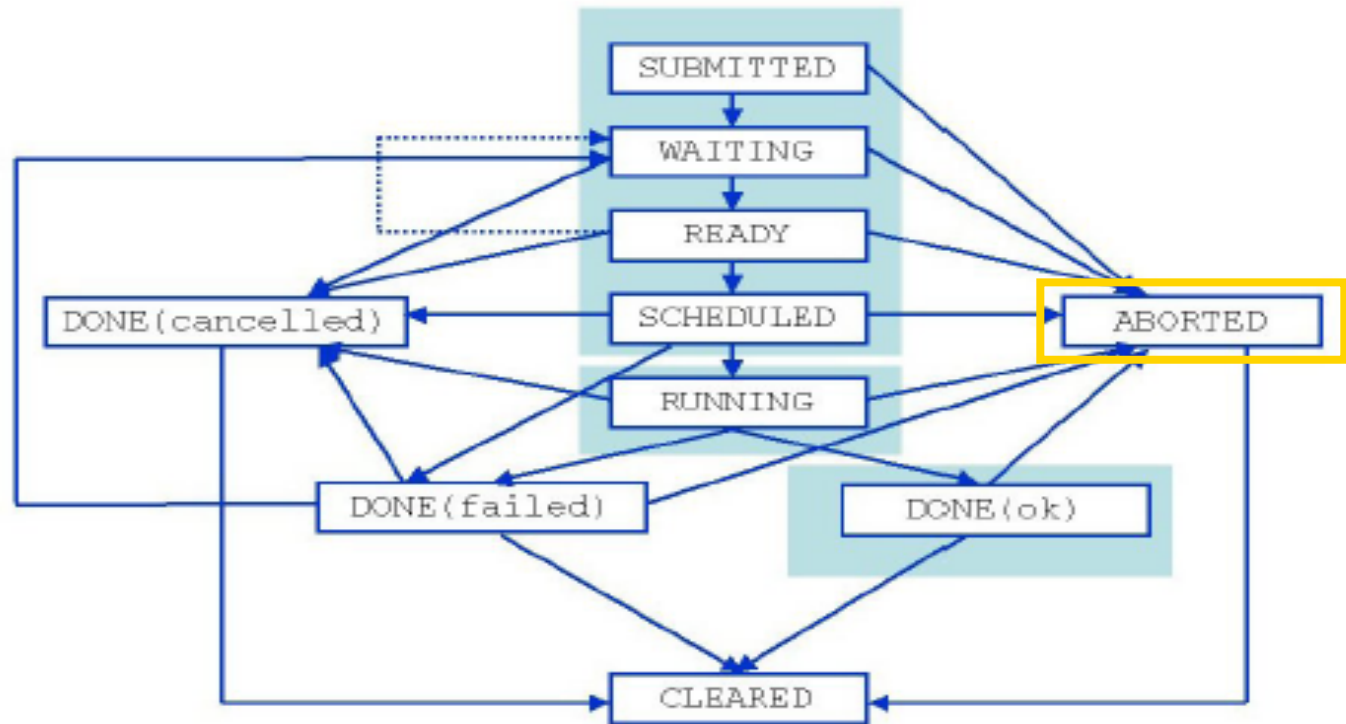
Running
job is
running.



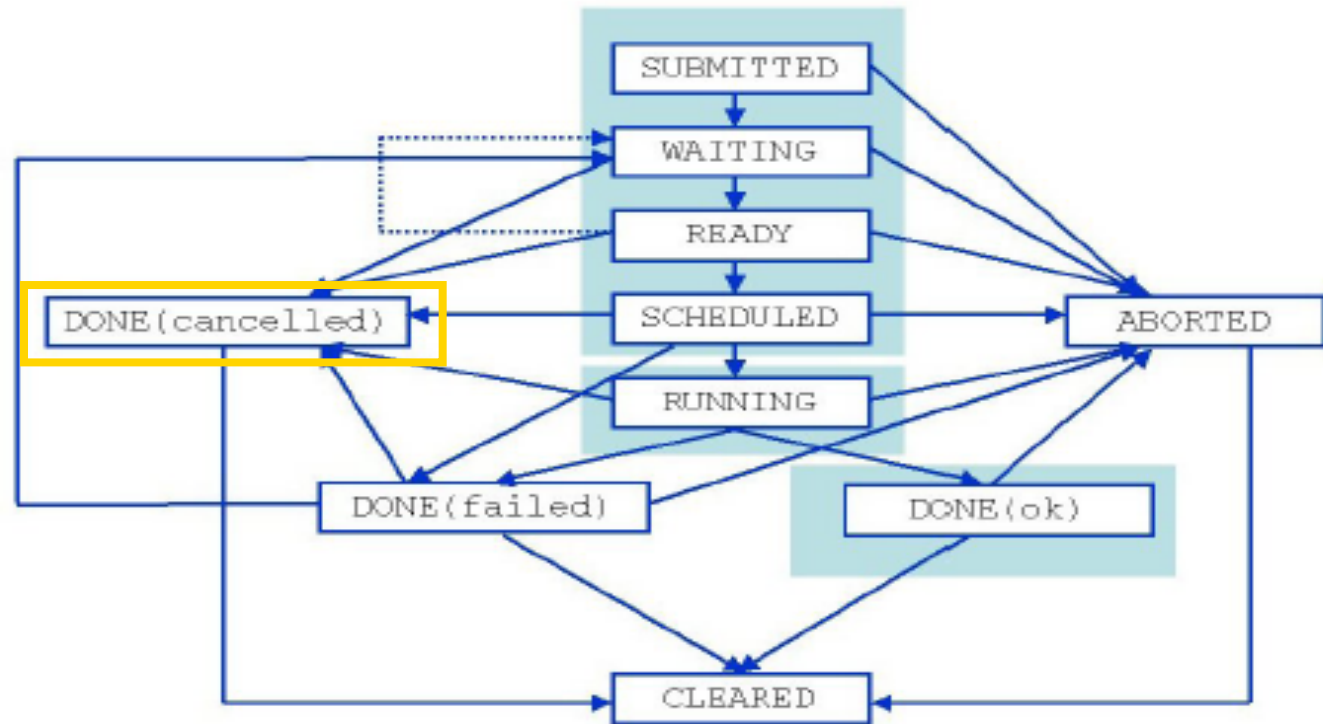
Done job exited or considered to be in a terminal state by CondorC (e.g., submission to CE has failed in an unrecoverable way).



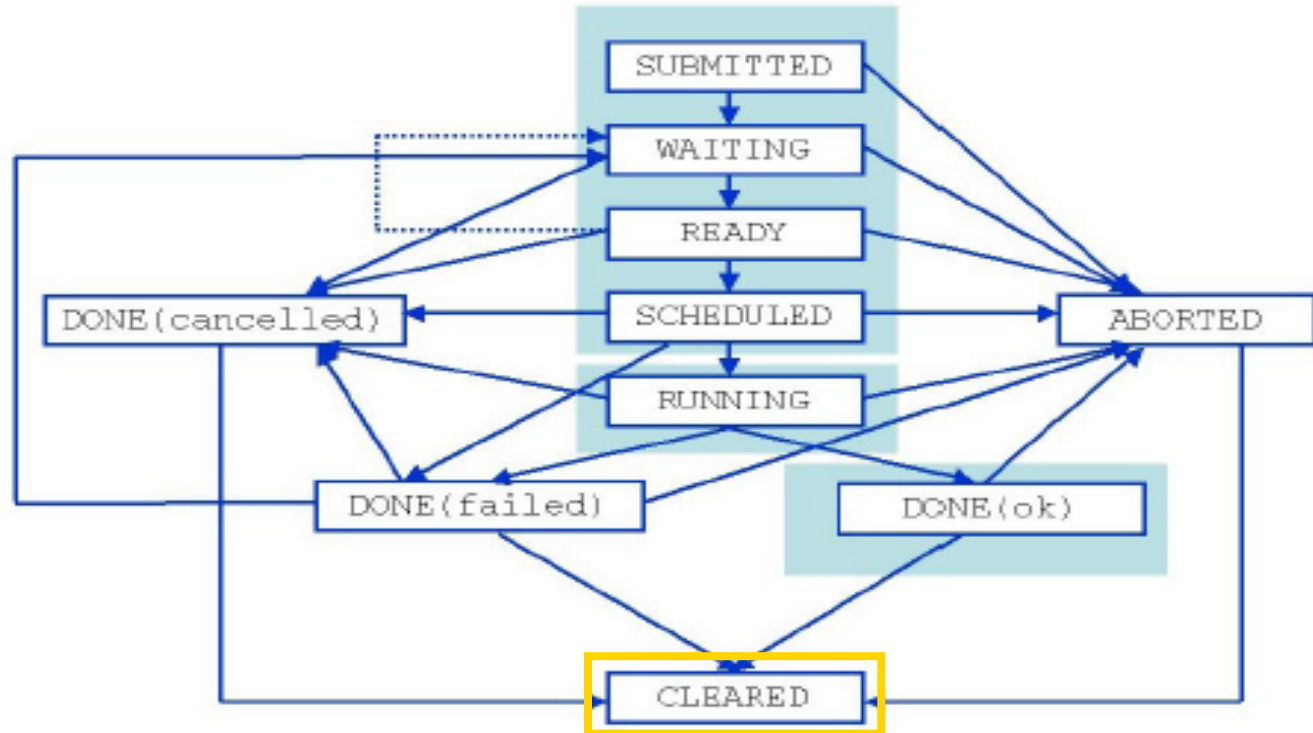
Aborted job processing was aborted by WMS (waiting in the WM queue or CE for too long, expiration of user credentials).



Cancelled job has been successfully canceled on user request.



Cleared output sandbox was transferred to the user or removed due to the timeout.



- **Every step of the job life cycle is logged on a service called Logging and Bookkeeping**
- **It is useful for users willing to know the status of their execution**
 - when a job is submitted the UI logs it on LB
 - WMS logs each step of scheduling
 - CE logs when it receive a job (scheduled), when it's running and when it's done

- The CE is the front-end machine (master node) to a local batch system
 - supported batch systems are PBS(Torque/MAUI), LSF, Condor
- WMS “pushes” job execution requests to the CE using **condor-G**
 - when a CE receives a job, this is moved on a queue
 - Then the job will be executed on the first available among its **Worker Nodes** (where the batch system clients run)
 - when execution is complete, output files are copied to the CE using scp
- If the job is successfully executed, output files are copied back to the WMS using **globus-url-copy**
- By queries to the LB, users know when a job is done and they can retrieve the output

- **WMS catches users' request for job executions**
- **Requests are expressed through JDL**
 - JDL allows to specify requirements that selected resources must have
- **The WMS processes request and chooses (matchmaking) a Computing Element for the actual execution**
 - Status of resources is known to WMS with queries to BDII
- **The CE tries to execute the job and copies back output files to WMS**
 - status of execution is logged on LB
- **Users queries LB, discovers their job is done and download output files from WMS**

- **Practice**

- <https://grid.ct.infn.it/twiki/bin/view/GILDA/SimpleJobSubmission>
- <https://grid.ct.infn.it/twiki/bin/view/GILDA/MoreOnJDL>
- <https://grid.ct.infn.it/twiki/bin/view/GILDA/JobDataWMS>

- **Deepen**

- <https://edms.cern.ch/file/674643/1/WMPROXY-guide.pdf>
- <https://edms.cern.ch/file/590869/1/EGEE-JRA1-TEC-590869-JDL-Attributes-v0-9.pdf>