gLite Job Management

Manuel Rodríguez-Pascual

Manuel.rodriguez -at- ciemat.es

CIEMAT, Madrid (Spain)

Latin America 3 2010 - Joint CHAIN/GISELA/EPIKH
School for Application Porting
Nov 29-Dec 9, 2010. Valparaíso(Chile)

Outline

- gLite architecture (short reminder)
- The WMS system
- The Job Description Language
- Examples

Key Elements

- The ***Workload Management System (WMS)*** is the gLite component that allows users to submit ***jobs***, and performs all tasks required to execute them, without exposing the user to the complexity of the Grid.
- ***Job Description Language (JDL)*** is the language used to describe a job. User have to describe his jobs and their requirements, and to retrieve the output when the jobs are finished.
- ***The Command Line Interface*** is a suite of gLite commands used in order to interact with the WMS.

Workload Management System

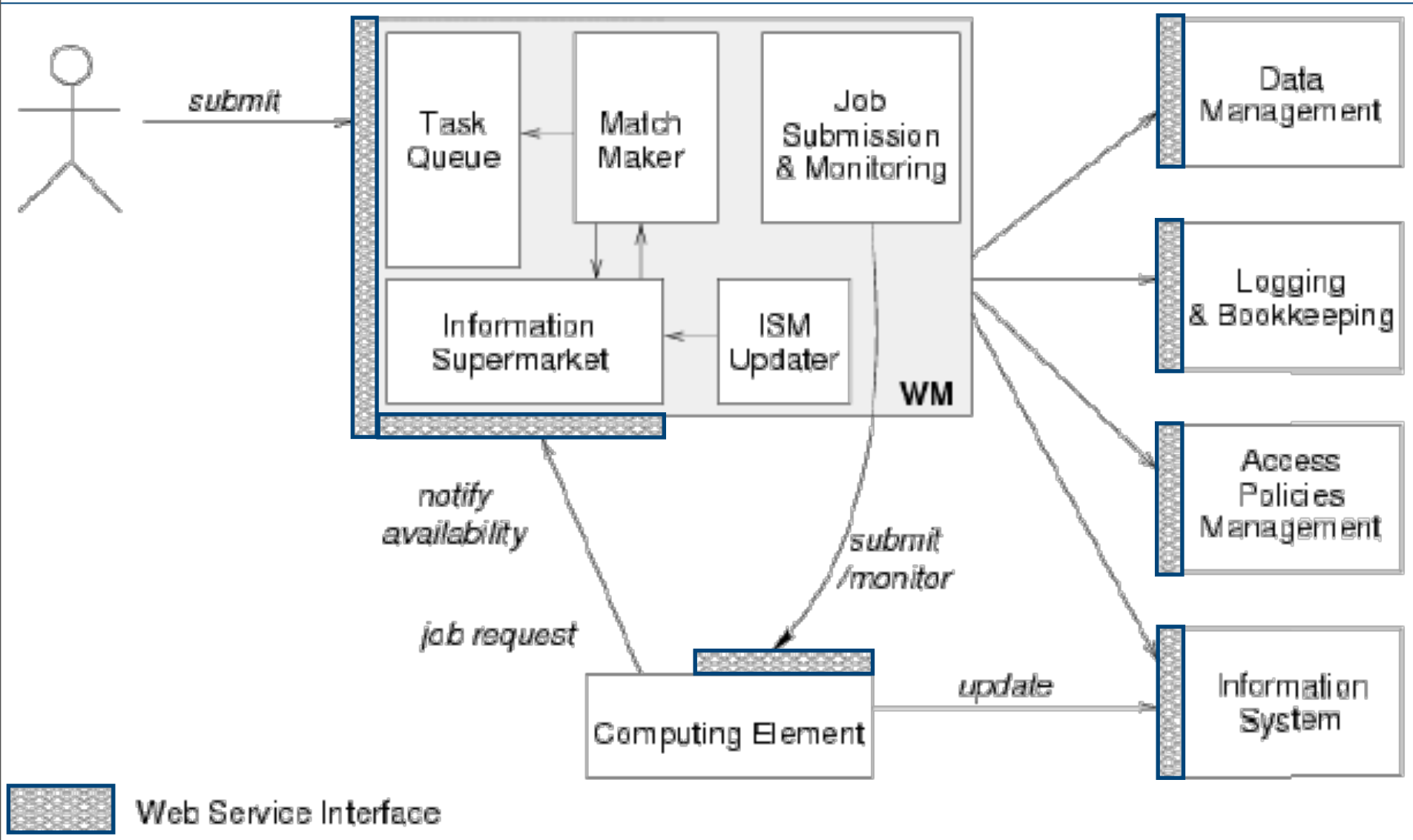
- The Workload Management System (WMS) consists of a set of Grid middleware components in charge of **distributing** and **managing** jobs across Grid resources

- **Purpose** of the Workload Manager (WM) is to accept and satisfy requests for job management coming from its clients

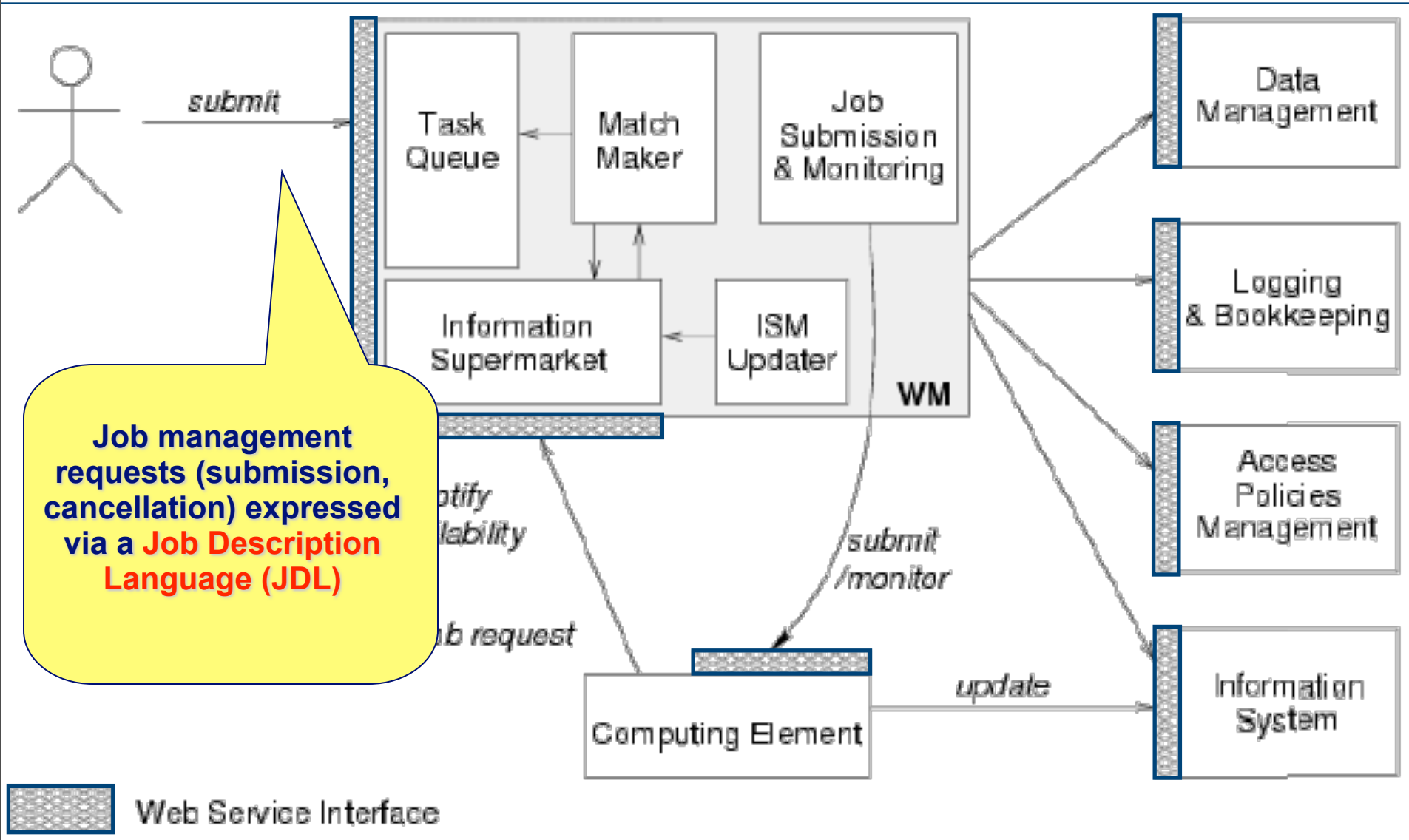
- The WM hands over the job to an appropriate **Computing Element** (CE) for execution taking into account requirements and the preferences expressed in the job description.

- The decision on which resource should be used is the outcome of the so called **matchmaking process**:
  - requests from the users for their jobs are matched against the available resources and their features

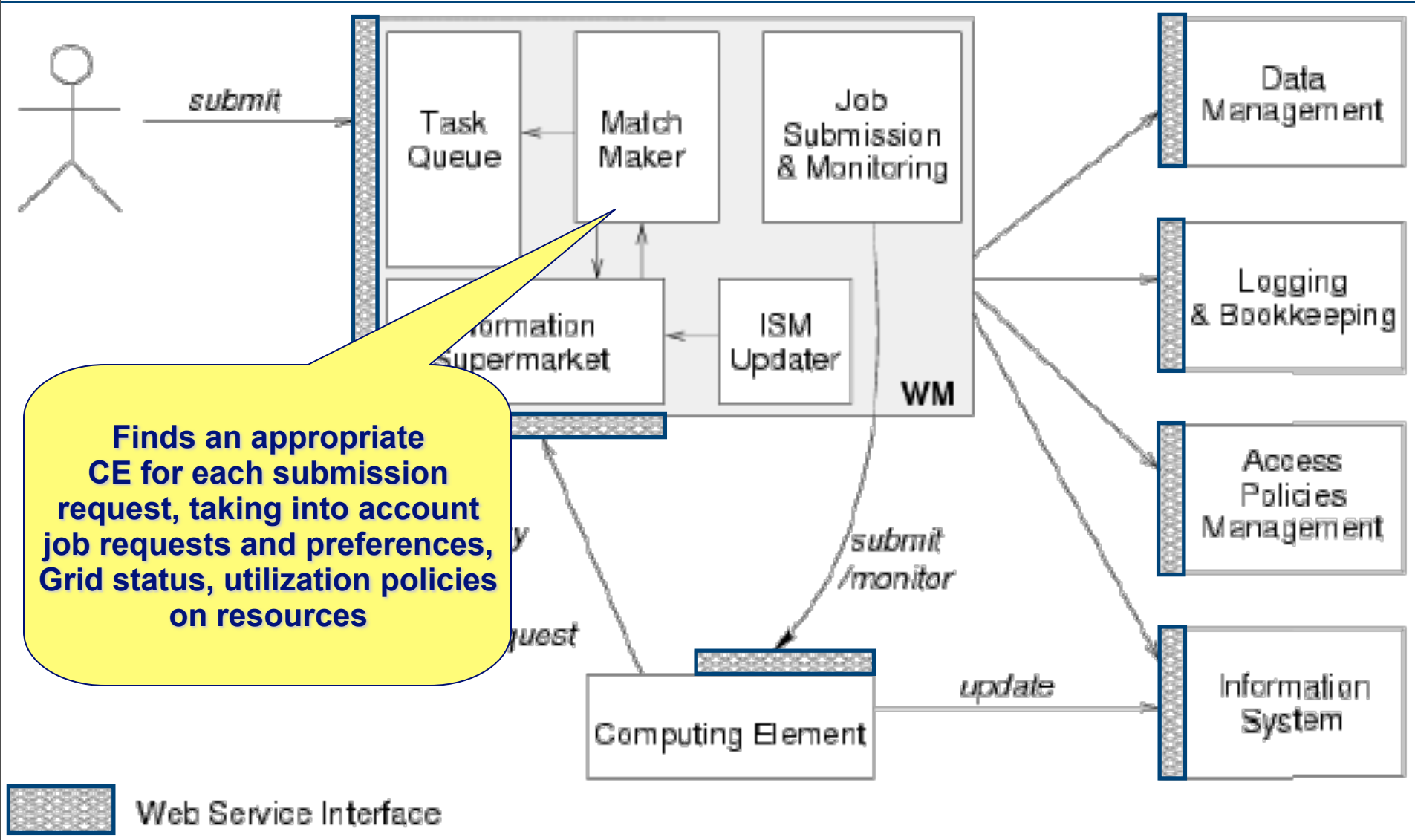
gLite WMS Architecture



gLite WMS Architecture

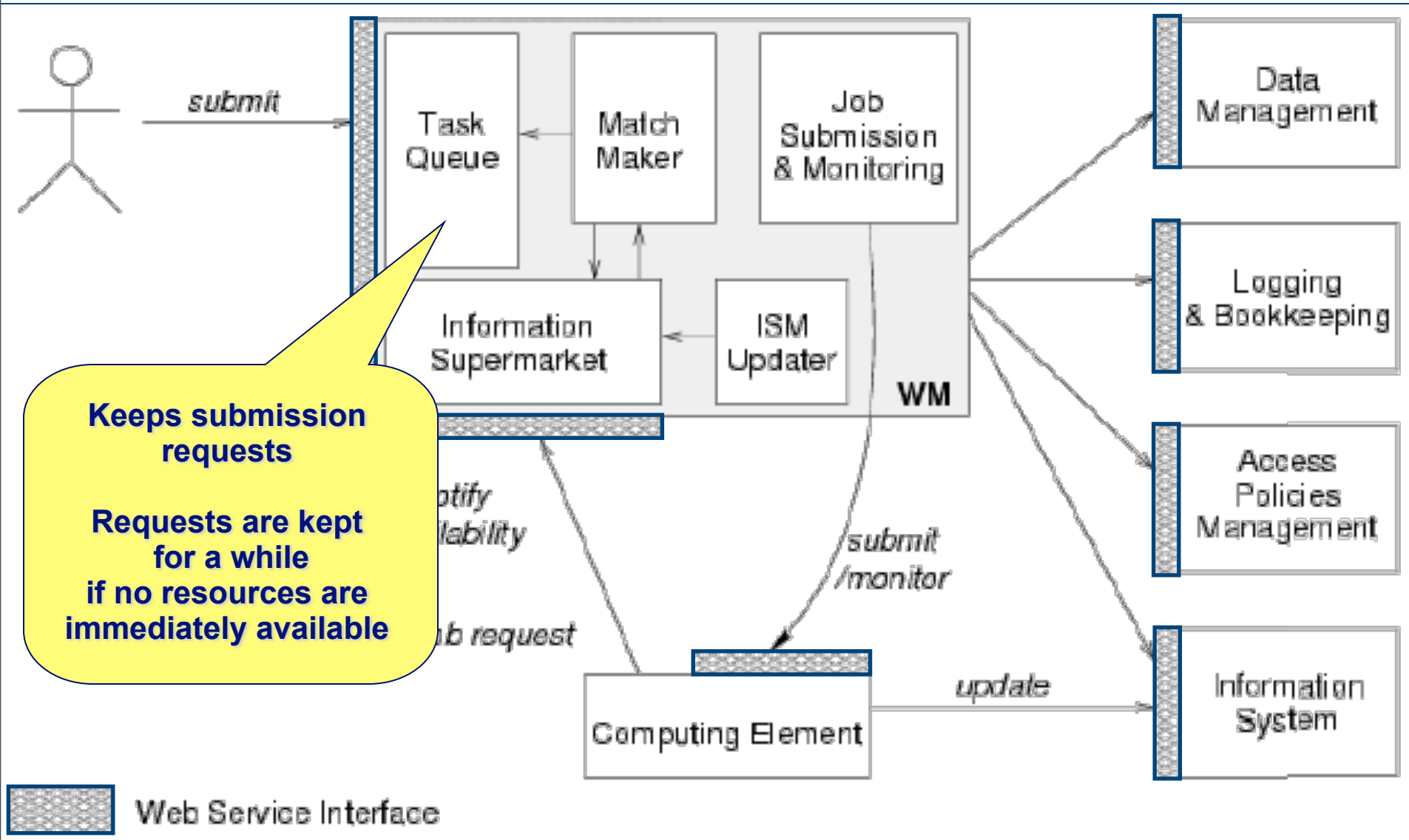


gLite WMS Architecture

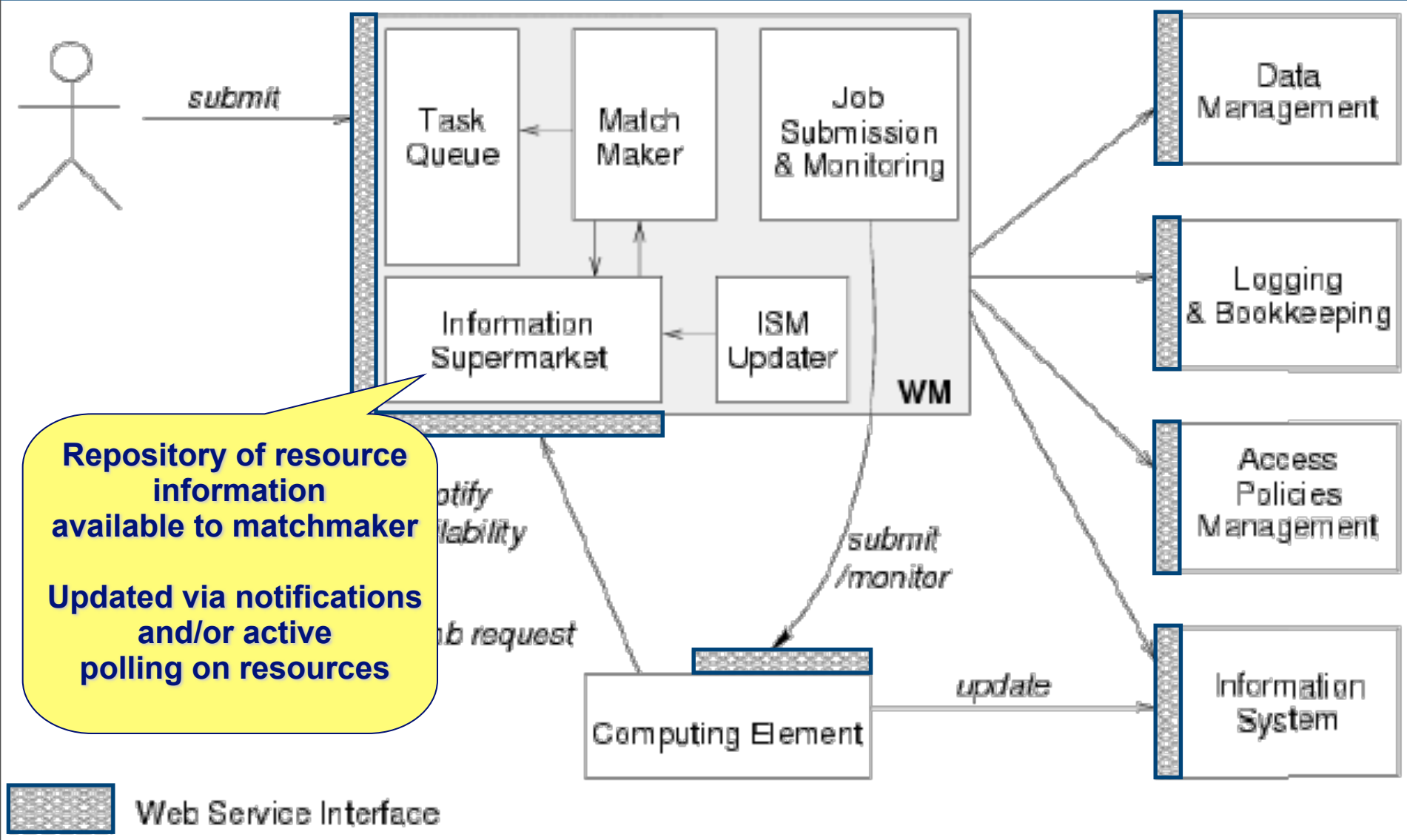


Finds an appropriate CE for each submission request, taking into account job requests and preferences, Grid status, utilization policies on resources

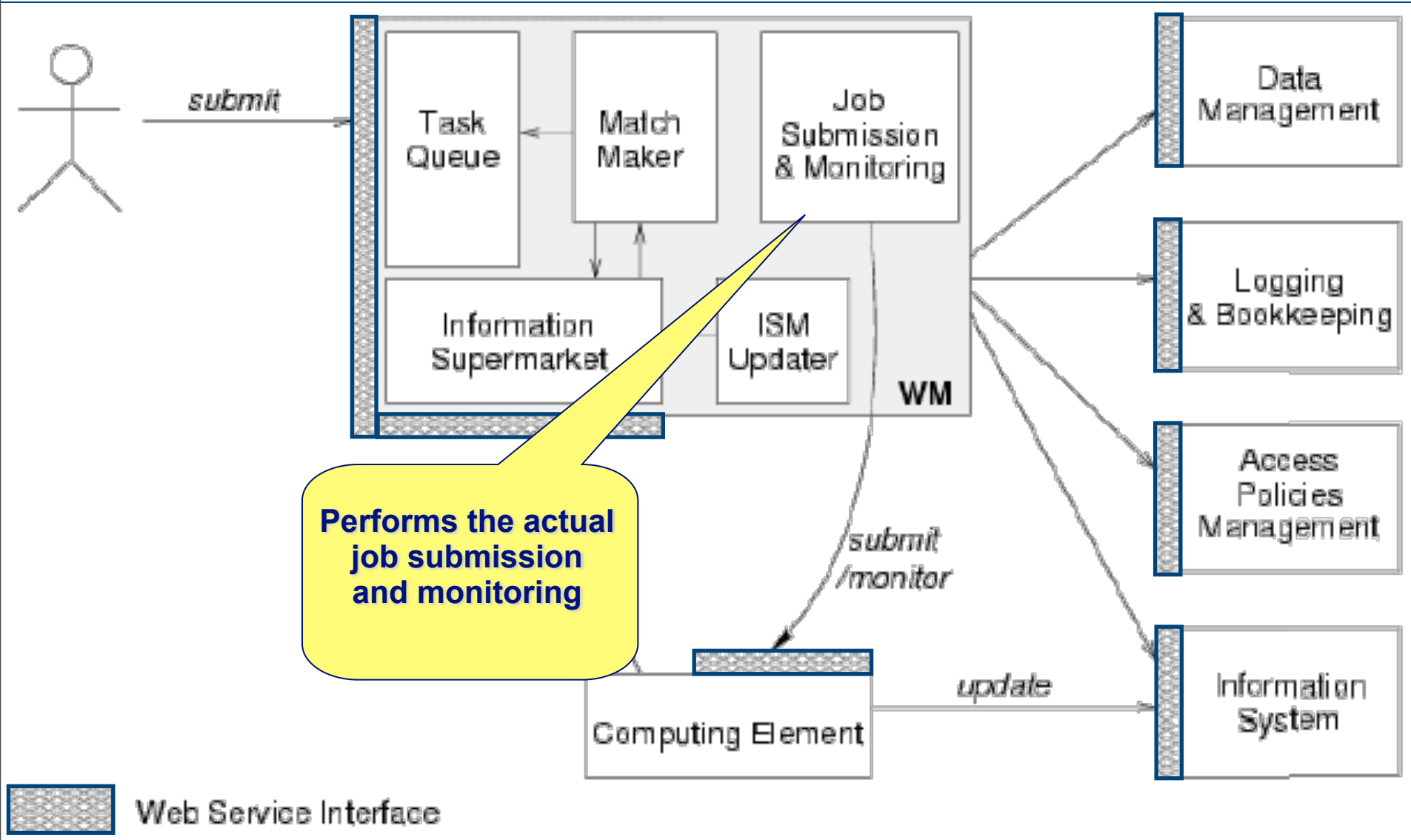
gLite WMS Architecture



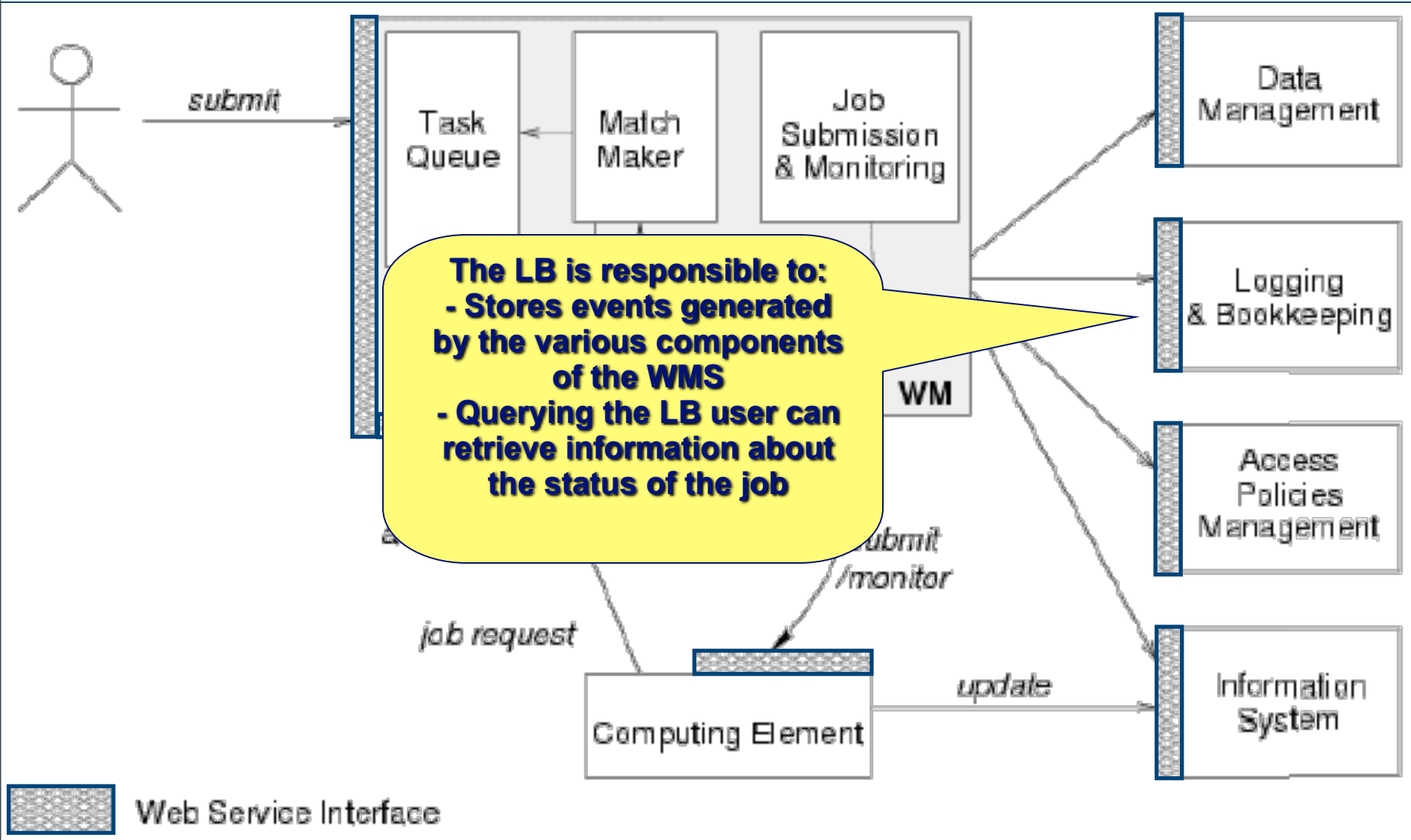
gLite WMS Architecture



gLite WMS Architecture



gLite WMS Architecture



Job Description Language

- The Job Description Language (JDL) is a high-level language, used to describe jobs and aggregates of jobs with arbitrary dependency relations.
  - A job description is a file (called **JDL file**) consisting of lines having the format: **`attribute = expression;`**

Job Description Language

- **The character “ ‘ ” cannot be used in the JDL.**
- Litteral Strings are enclosed in double quotes (“)
- Comments must be preceded by a sharp character (**#**) or a double slash (//) at the beginning of each line.
- Multi-line comments must be enclosed between “/*” and “*/” .

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

Simple example

```
Type = "Job";
JobType = "Normal";
Executable = "/bin/hostname";
StdOutput = "hostname.out";
StdError = "hostname.err";
OutputSandbox = {"hostname.err","hostname.out"};
Arguments = "-f";
```

Executable

```
Executable = "/bin/hostname";
```

The Executable attribute specifies the command to be run by the job. If the command is already present on the WN, it must be expressed as a **absolute path**; if it has to be copied from the UI, only the file name must be specified, and the path of the command on the UI should be given in the **InputSandbox** attribute.

```
Executable = "test.sh";
InputSandbox = {"/home/doe/
test.sh"};
```

Arguments

- The Arguments attribute can contain a string value, which is taken as argument list for the executable:

```
Executable = "/bin/hostname";
Arguments = "-f";
```

- In the **Executable** and in the **Arguments** attributes it may be necessary to use special characters, such as **&**, ****, **|**, **>**, **<**. These characters should be preceded by triple \ in the JDL, or specified inside quoted strings e.g.:

```
Arguments = "-f file1\\\&file2";
```

StdOutput and StdError

- The attributes StdOutput and StdError define the name of the files containing the standard output and standard error of the executable, once the job output is retrieved.

```
StdOutput = "std.out";
StdError = "std.err";
```

InputSandbox and OutSandbox

- If files have to be copied from the UI to the execution node, they must be listed in the InputSandbox attribute:

```
InputSandbox = {"test.sh", .. , "fileN"};
```

- The files to be transferred back to the UI after the job is finished can be specified using the **OutputSandbox** attribute:

```
OutputSandbox = {"std.out","std.err"};
```

- **!** The InputSandbox cannot contain two files with the same name, even if they have a different absolute path, as when transferred they would overwrite each other.

Enviroment and Virtual Organisation

- The shell environment of the job can be modified using the **Environment** attribute.

```
Environment = {"CMS_PATH=$HOME/cms",
"CMS_DB=$CMS_PATH/cmdb"};
```

- The **VirtualOrganisation** attribute can be used to explicitly specify the VO of the user:

```
VirtualOrganisation = "gilda";
```

JobType

- **JobType**
  - ***Normal*** **(simple, sequential job),** ***Interactive*****,** ***MPI*****,** ***Checkpointable*****,** ***Partitionable, Parametric***
  - Or combination of them
    - Checkpointable, Interactive
    - Checkpointable, MPI

```
JobType = "Interactive";
```

! `JobType = {"Interactive","Checkpointable"};`

"Interactive" + "MPI" not yet permitted

Requirements

- The **Requirements** attribute can be used to express constraints on the resources where the job should run.
  - **Its value is a Boolean expression that must evaluate to true for a job to run on that specific CE.**

- **Note:** *Only one Requirements attribute can be specified* (if there are more than one, only the last one is considered).If several conditions must be applied to the job, then they all must be combined in a single Requirements attribute.

- For example, let us suppose that the user wants to run on a CE using PBS as batch system, and whose WNs have at least two CPUs. He will write then in the job description file:

```
Requirements = other.GlueCEInfoLRMSType == "PBS" &&
    other.GlueCEInfoTotalCPUs > 1;
```

Requirements (Cont.)

- The WMS can be also asked to send a job to a particular queue in a CE with the following expression:

```
Requirements = other.GlueCEUniqueID == "lxshare0286.cern.ch:
    2119/jobmanager-pbs-short";
```

- It is also possible to use regular expressions when expressing a requirement.
  - **Let us suppose for example that the user wants all his jobs to run on any CE in the domain cern.ch. This can be achieved putting in the JDL file the following expression:**

```
Requirements = RegExp("cern.ch",other.GlueCEUniqueID);
```

- The opposite can be required by using:

```
Requirements = (!RegExp("cern.ch",
other.GlueCEUniqueID));
```

RetryCount

- If the job duration is significant, it is strongly advised to put a requirement on the maximum CPU time, or the wallclock time (expressed in minutes), needed for the job to complete.
  - **For example, to express the fact that the job needs at least 8 CPU hours and 20 wallclock hours:**

```
Requirements = other.GlueCEPolicyMaxCPUTime > 480 &&
    other.GlueCEPolicyMaxWallClockTime > 720;
```

- It is possible to have the WMS automatically resubmitting jobs which, for some reason, are aborted by the Grid. The user can limit the number of times the WMS should resubmit a job by using the JDL attributes **`RetryCount`**.

```
RetryCount = 7; or RetryCount = 0;
```

Rank

- The choice of the CE where to execute the job, among all the ones satisfying the requirements, is based on the ***Rank*** of the CE, a quantity expressed as a floating-point number. The CE with the highest rank is the one selected.
  - **By default, the rank is equal to `other.GlueCEStateEstimatedResponseTime`, where the estimated response time is an estimation of the time interval between the job submission and the beginning of the job execution.**
  - `Rank = other.GlueCEStateFreeCPUs;`
    which will rank the best CE with the most free CPUs.

The Command Line Interface

- The gLite WMS used to implement two different services to manage jobs: the ***Network Server*** and the ***WMProxy***.
  - The only method to manage TODAY jobs is through the gLite WMS via WMProxy, because it gives the best performance and allows to use the most advanced functionalities

    (the network server is now obsolete)

- The WMProxy implements several new functionalities, among which:
  - **submission of job collections**
  - **faster authentication**
  - **faster match-making**
  - **faster response time for users**
  - **higher job throughput**

Creating the Proxy

- **$ voms-proxy-init --voms gilda**

```
Enter GRID pass phrase: VALPARAISO
Your identity: /C=IT/O=GILDA/OU=Personal Certificate/L=VALPARAISO/
    CN=VALPARAISO06
Creating temporary
    proxy
    ..................................................................................
    ..... Done
Contacting  voms.ct.infn.it:15001 [/C=IT/O=INFN/OU=Host/L=Catania/
    CN=voms.ct.infn.it] "gilda" Done
Creating proxy ..................................................................
    Done
Your proxy is valid until Mon Nov 29 22:41:39 2010
```

Submitting a simple job

- Starting from a simple JDL file, we can submit it via WMProxy by doing:

```
$ glite-wms-job-submit -a hostname.jdl
Connecting to the service https://rb-eugrid.eri.sci.eg:7443/
    glite_wms_wmproxy_server
==================== glite-wms-job-submit Success ============
The job has been successfully submitted to the WMProxy
Your job identifier is:

https://rb-eugrid.eri.sci.eg:9000/YS66UAF3cR5Ih65dPVKI2Q

==============================================================
```

Submitting a simple job

- The command returns to the user the **job identifier (*jobID*)**, which uniquely defines the job and can be used to perform further operations on the job, like interrogating the system about its status, or canceling it.

- The format of the jobID is:

```
https://<LB_hostname>[:<port>]/
    <unique_string>
```

- where **<unique string>** is guaranteed to be unique and **<LB hostname>** is the host name of the Logging and Bookkeeping (LB) server for the job, which usually sits on the WMS used to submit the job.

glite-wms-job-submit Options

- The **-o <file path>** option allows users to specify a file to which the jobID of the submitted job will be appended. This file can be given to other job management commands to perform operations on more than one job with a single command, and it is a convenient way to keep trace of one's jobs.

```
$ glite-wms-job-submit -d mydelegID -o jobid test.jdl
```

- The **-r <CEId>** option is used to directly send a job to a particular CE. If used, the match making will not be carried out.
  - **The drawback is that the *BrokerInfo* file, which provides information about the evolution of the job, will not be created, and therefore the use of this option is discouraged.**

```
$ glite-wms-job-submit -d mydelegID -r <CEId>
                       test.jdl
```

Listing CE(s) that matching a job

- It is possible to see which CEs are useful to run a job described by a given JDL using:

```
$ glite-wms-job-list-match a --rank hostname.jdl
Connecting to the service https://gilda-wms-01.ct.infn.it:7443/glite_wms_wmproxy_server

==========================================================================

                        COMPUTING ELEMENT IDs LIST
The following CE(s) matching your job requirements have been found:

                *CEId*
- ce.hpc.iit.bme.hu:2119/jobmanager-lcgpbs-gilda
- ce.scope.unina.it:2119/jobmanager-lcgpbs-egee_long
- ce.scope.unina.it:2119/jobmanager-lcgpbs-egee_short
- ce1-egee.srce.hr:2119/jobmanager-sge-prod
- gilda-01.pd.infn.it:2119/jobmanager-lcgpbs-gilda
- gn0.hpcc.sztaki.hu:2119/jobmanager-lcgpbs-gilda
- grid010.ct.infn.it:2119/jobmanager-lcgpbs-gilda
- grisuce.scope.unina.it:2119/jobmanager-lcgpbs-grisu_long
- grisuce.scope.unina.it:2119/jobmanager-lcgpbs-grisu_short
- iceage-ce-01.ct.infn.it:2119/jobmanager-lcgpbs-gilda
- sirius-ce.ct.infn.it:2119/jobmanager-lcgpbs-gilda
```

Retrieving the status of a job

```
$ glite-wms-job-status https://rb-eugrid.eri.sci.eg:9000/yuqCRO-BS1Bek8uZ7srDYw

======================== glite-wms-job-status Success =====================
BOOKKEEPING INFORMATION:

Status info for the Job : https://rb-eugrid.eri.sci.eg:9000/yuqCRO-BS1Bek8uZ7srDYw
Current Status:     Done (Success)
Logged Reason(s):
    - job completed
    - Job Terminated Successfully
Exit code:          0
Status Reason:      Job Terminated Successfully
Destination:        srvslngrd004.uct.ac.za:8443/cream-pbs-eumed
Submitted:          Sat Oct 23 22:11:20 2010 EET
```

- The *verbosity* level controls the amount of information provided. The value of the **-v option** ranges from **0** to **3**.
- The commands to get the job status can have several jobIDs as arguments, or you can use the **-i <file path>** option:

```
glite-wms-job-status -i jobid
```

Cancelling a job

```
$ glite-wms-job-cancel https://rb-eugrid.eri.sci.eg:
    9000/vJDGfi2azmRQ45b80tmAfq
Are you sure you want to remove specified job(s) [y/n]y : y

Connecting to the service https://rb-eugrid.eri.sci.eg:7443/
    glite_wms_wmproxy_server
==================== glite-wms-job-cancel Success ================
The cancellation request has been successfully submitted for the
    following job(s):

- https://rb-eugrid.eri.sci.eg:9000/vJDGfi2azmRQ45b80tmAfg
============================================================
```

If the cancellation is successful, the job will terminate in status **CANCELLED**

Retrieving the output(s)

```
$ glite-wms-job-output https://rb-eugrid.eri.sci.eg:9000/
    yuqCRO-BS1Bek8uZ7srDYw

Connecting to the service https://rb-eugrid.eri.sci.eg:7443/
    glite_wms_wmproxy_server
================================================================

                        JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
https://rb-eugrid.eri.sci.eg:9000/yuqCRO-BS1Bek8uZ7srDYw
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/dessokey_yuqCRO-BS1Bek8uZ7srDYw
```

- The default location for storing the outputs (normally **/tmp**) is defined in the UI configuration, but it is possible to specify in which directory to save the output using the **--dir <path name>** option.

```
glite-wms-job-output -i jobId -dir /path
```

‘Scattered’ Input Sandboxes

- A new feature introduced by the gLite WMS is the possibility to indicate input sandbox files stored not on the UI,but on a GridFTP server, and, similarly, to specify that output files should be transferred to a GridFTP server when the job finishes.

```
InputSandbox = {"gsiftp://lxb0707.cern.ch/cms/fileA",
    "fileB"};
```

- It is also possible to specify a base GridFTP URI with the attribute **InputSandboxBaseURI**
  - **files expressed as simple file names or as relative paths will be looked for under that base URI.**

```
InputSandbox = {"fileA", "data/fileB", "file:///home/
    doe/fileC"};

InputSandboxBaseURI = "gsiftp://lxb0707.cern.ch/cms/
    doe";
```

Storing output files in a GridFTP Server

- In order to store the output sandbox files to a GridFTP server, the **OutputSandboxDestURI** attribute must be used together with the usual OutputSandbox attribute.
  - **The latter is used to list the output files created by the job in the WN to be transferred.**
  - **The former is used to express where the output files are to be transferred.**

```
OutputSandbox = {"fileA", "data/fileB", "fileC"};

OutputSandboxDestURI = {"gsiftp://lxb0707.cern.ch/cms/
    doe/fileA",
     "gsiftp://lxb0707.cern.ch/cms/doe/fileB","fileC"};
```

  - **where the first two files have to be copied to a GridFTP server, while the third file will be copied back to the WMS with the usual mechanism. Clearly, glite-wms-job-output will retrieve only the third file.**

- Another possibility is to use the **OutputSandboxBaseDestURI** attribute to specify a base URI on a GridFTP server where the files listed in OutputSandbox will be copied.

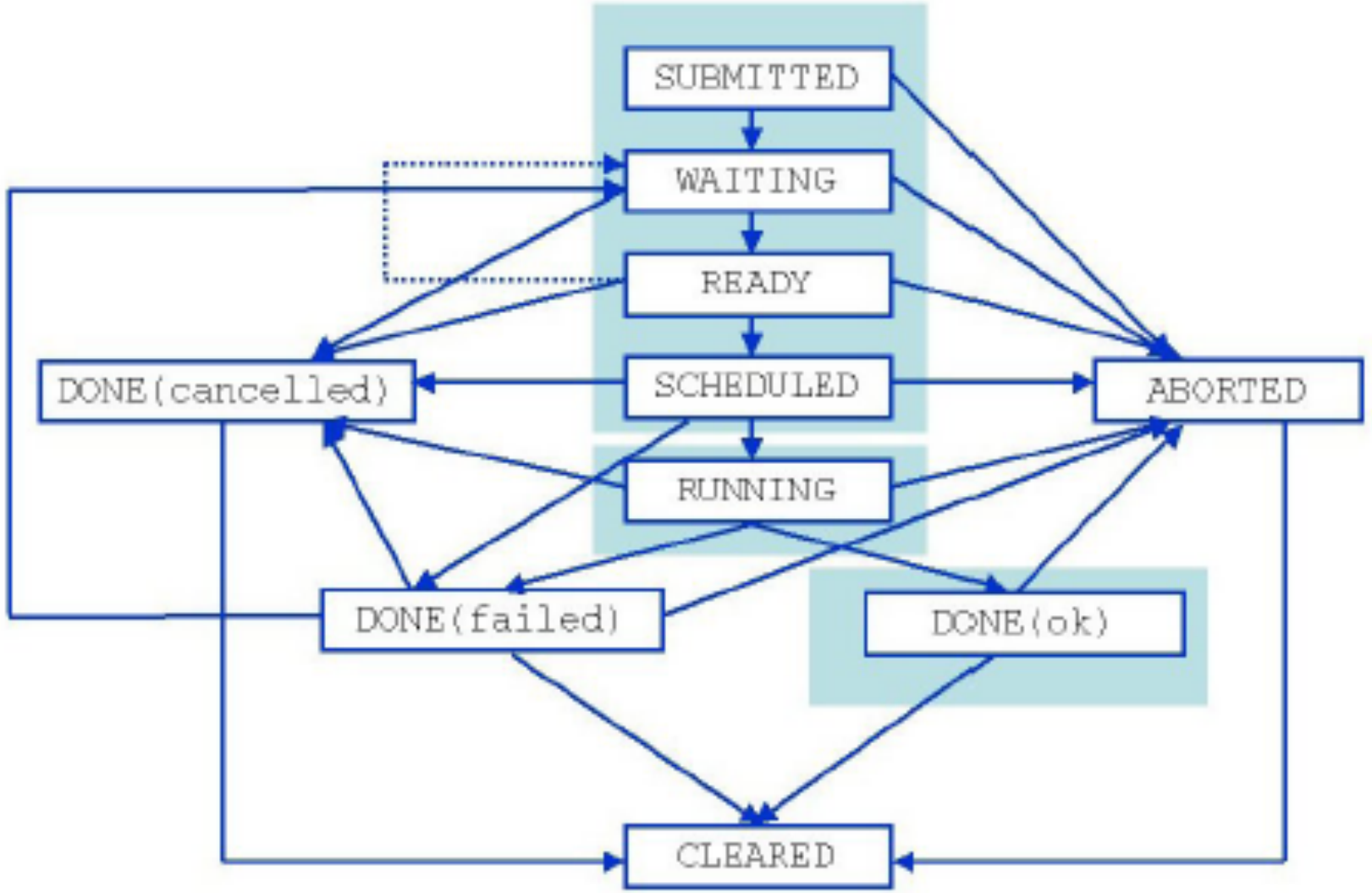
```
OutputSandbox = {"fileA", "fileB"};

OutputSandboxBaseDestURI = "gsiftp://lxb0707.cern.ch/
    cms/doe/";
```

will copy both files under the specified GridFTP URI.

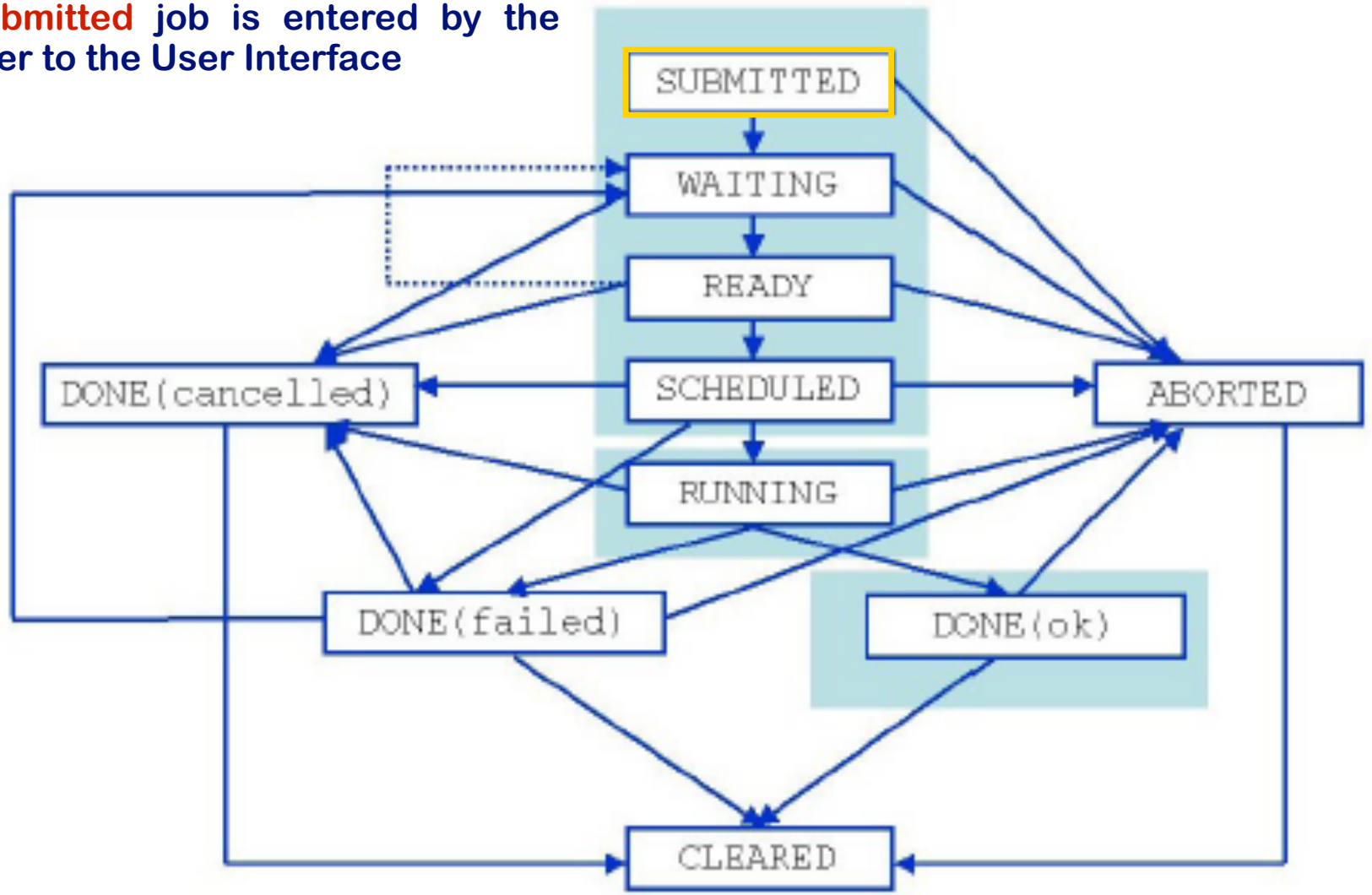
! **Note:** the directory on the GridFTP where the files have to be copied must already exist.

Jobs State Machine (1/9)

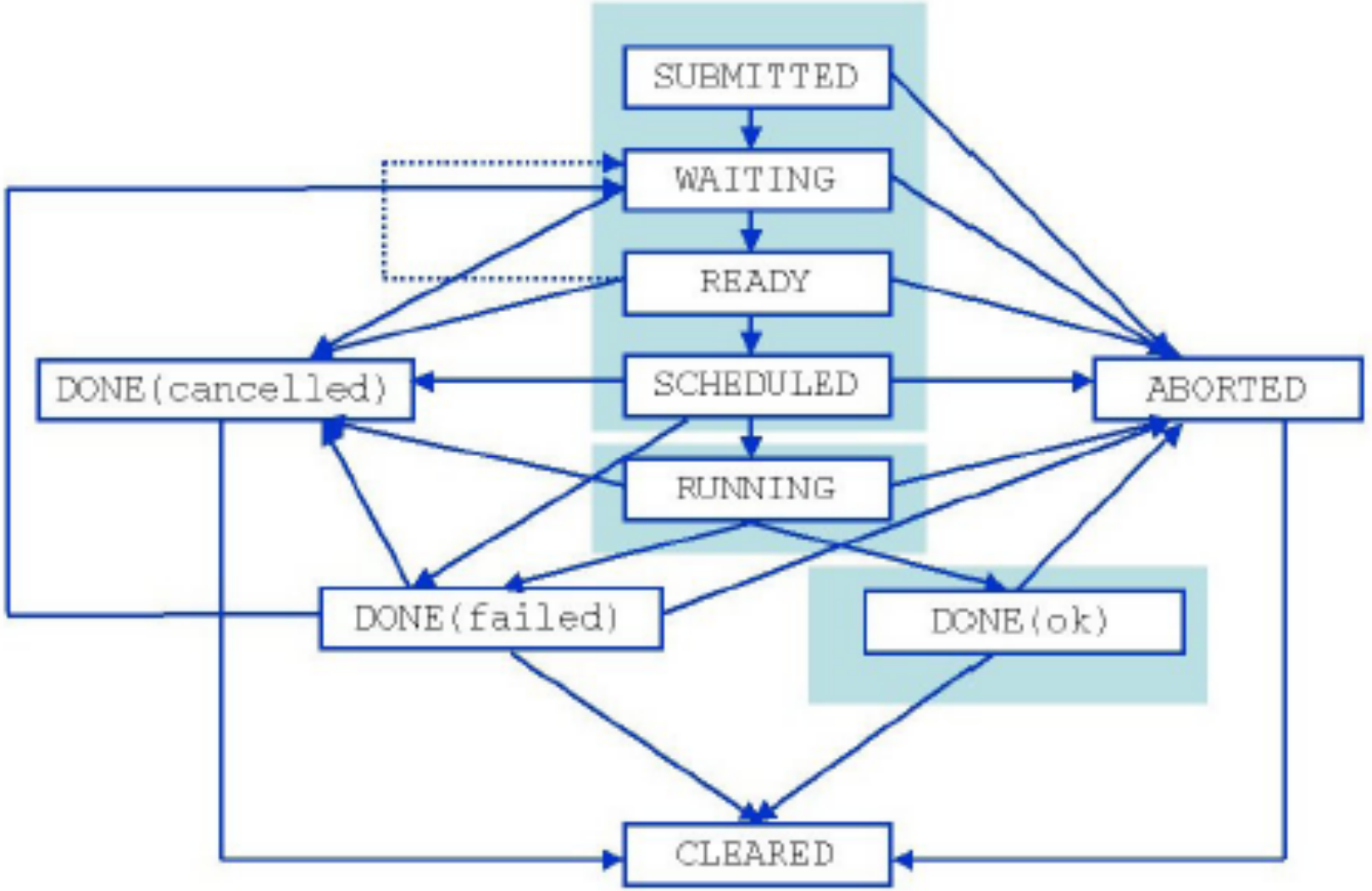


Jobs State Machine (1/9)

Submitted job is entered by the user to the User Interface

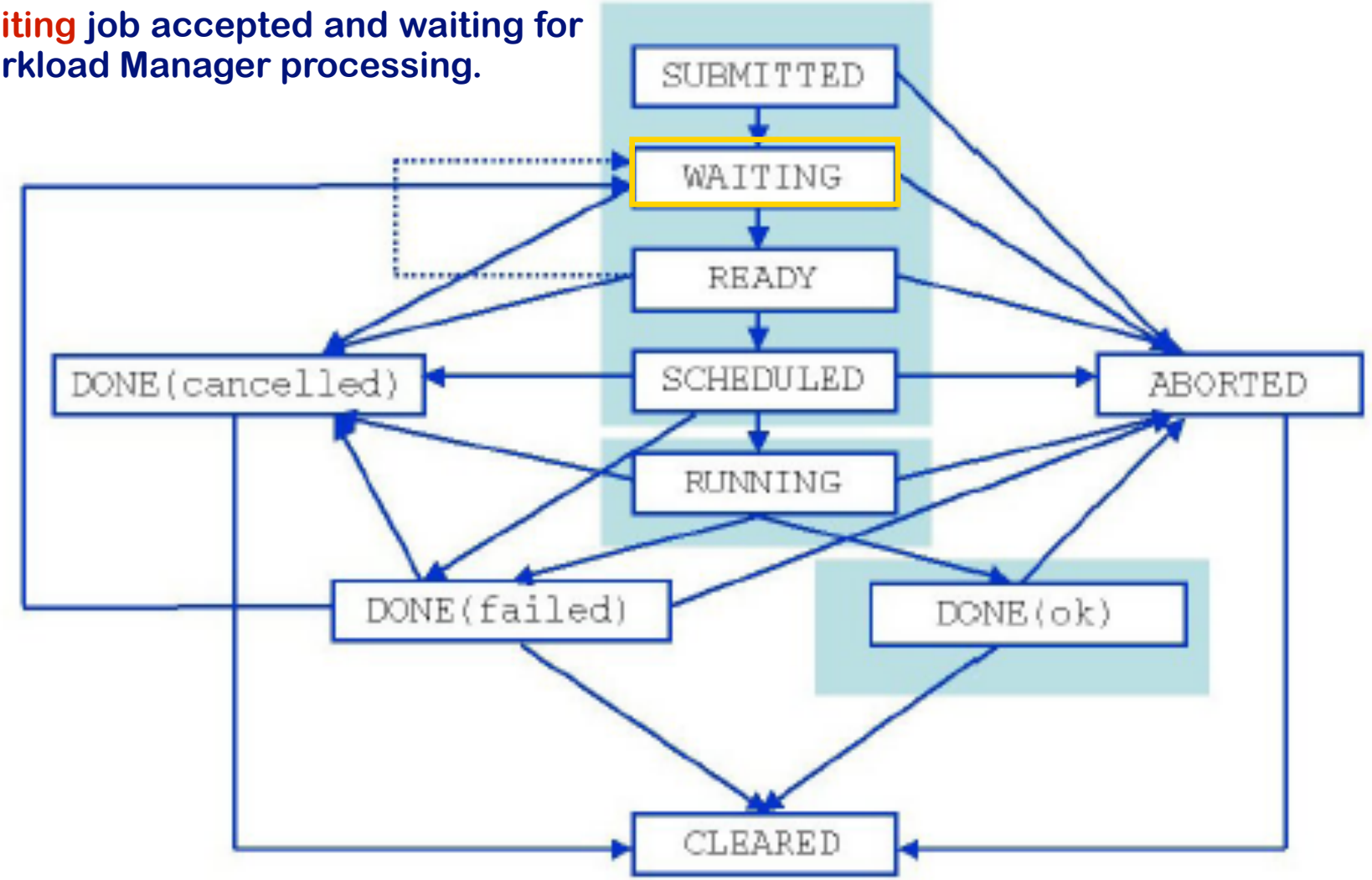


Jobs State Machine (2/9)

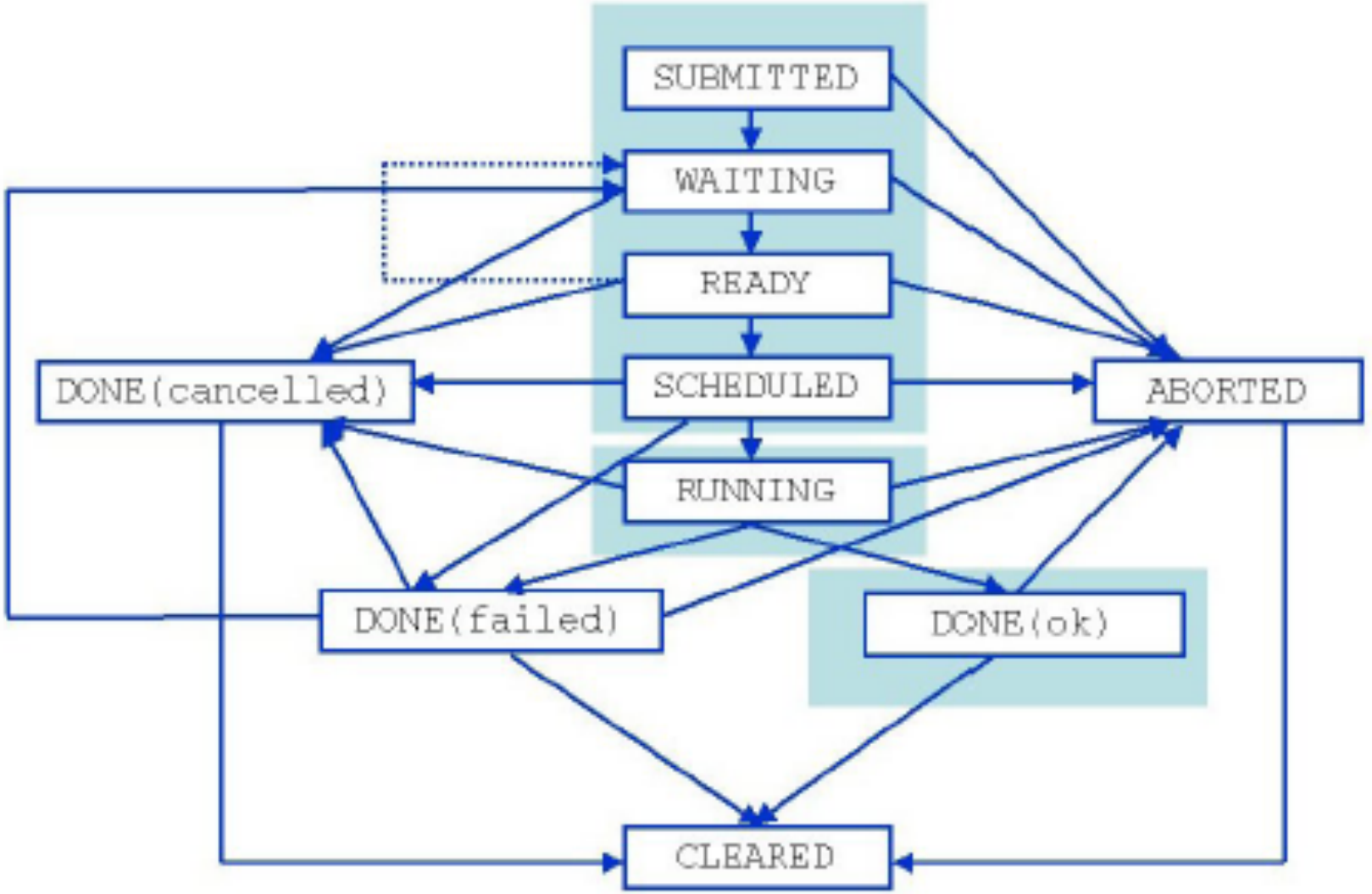


Jobs State Machine (2/9)

Waiting job accepted and waiting for Workload Manager processing.

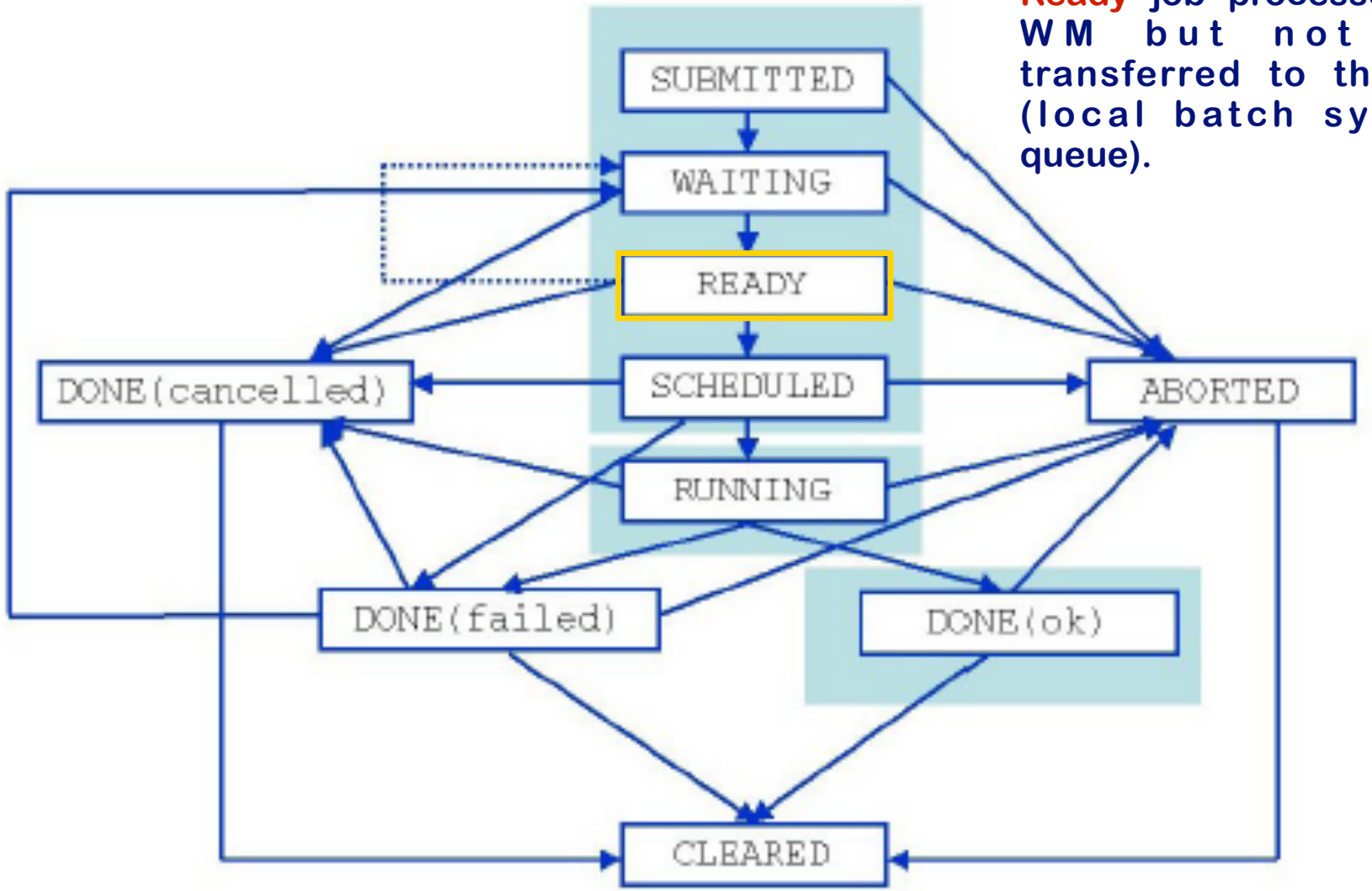


Jobs State Machine (3/9)

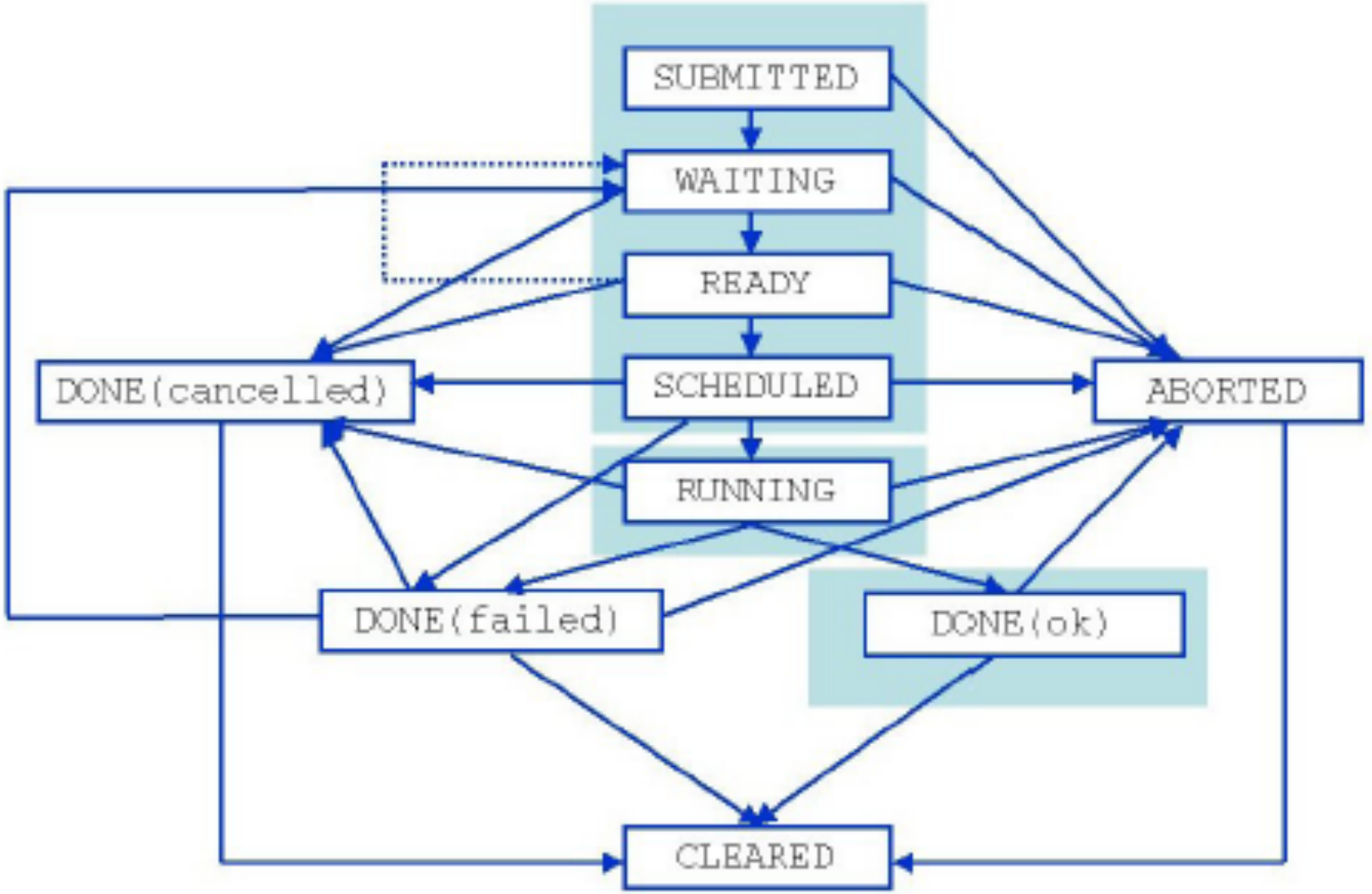


Jobs State Machine (3/9)

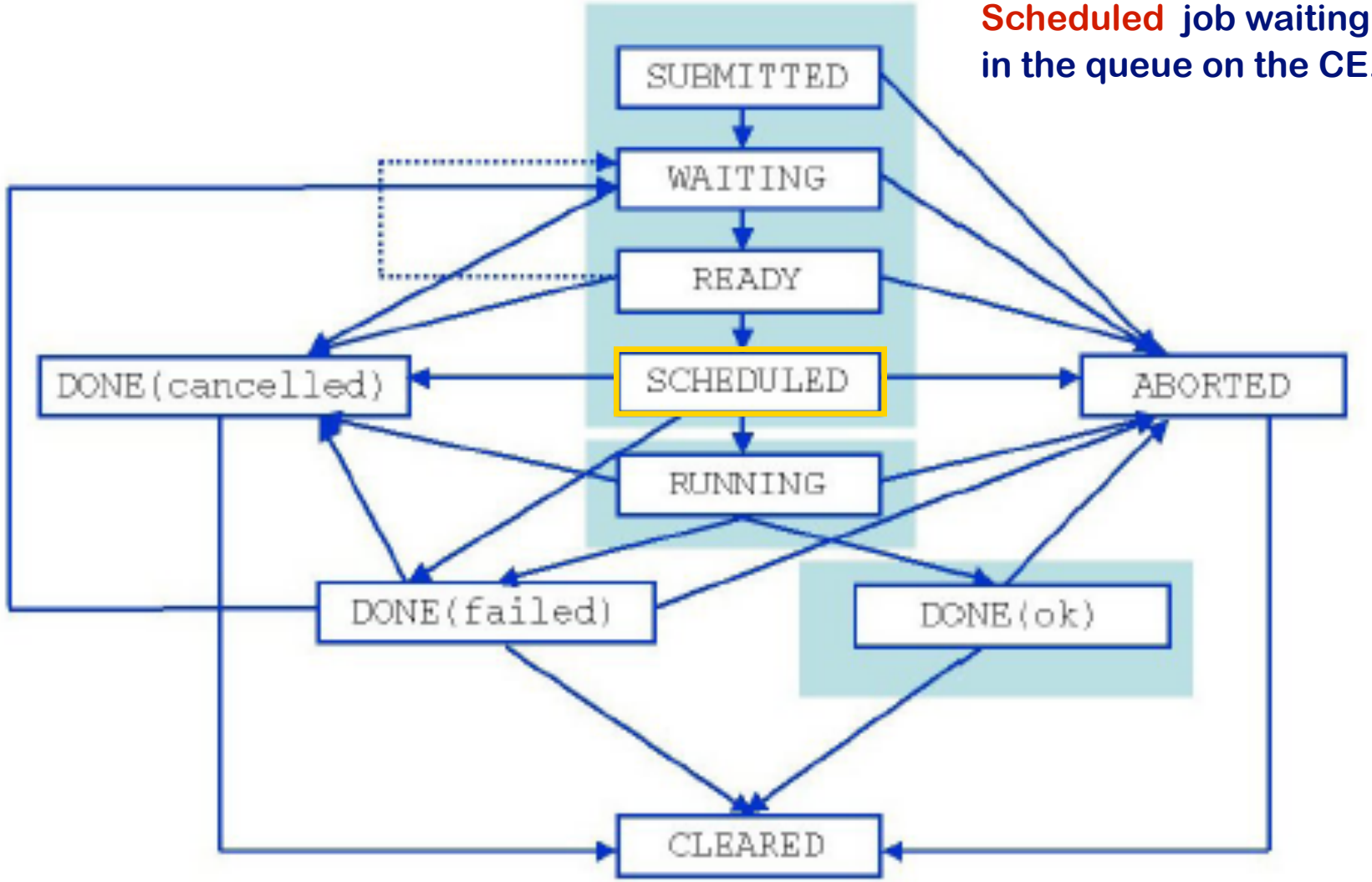
Ready job processed by WM but not yet transferred to the CE (local batch system queue).



Jobs State Machine (4/9)

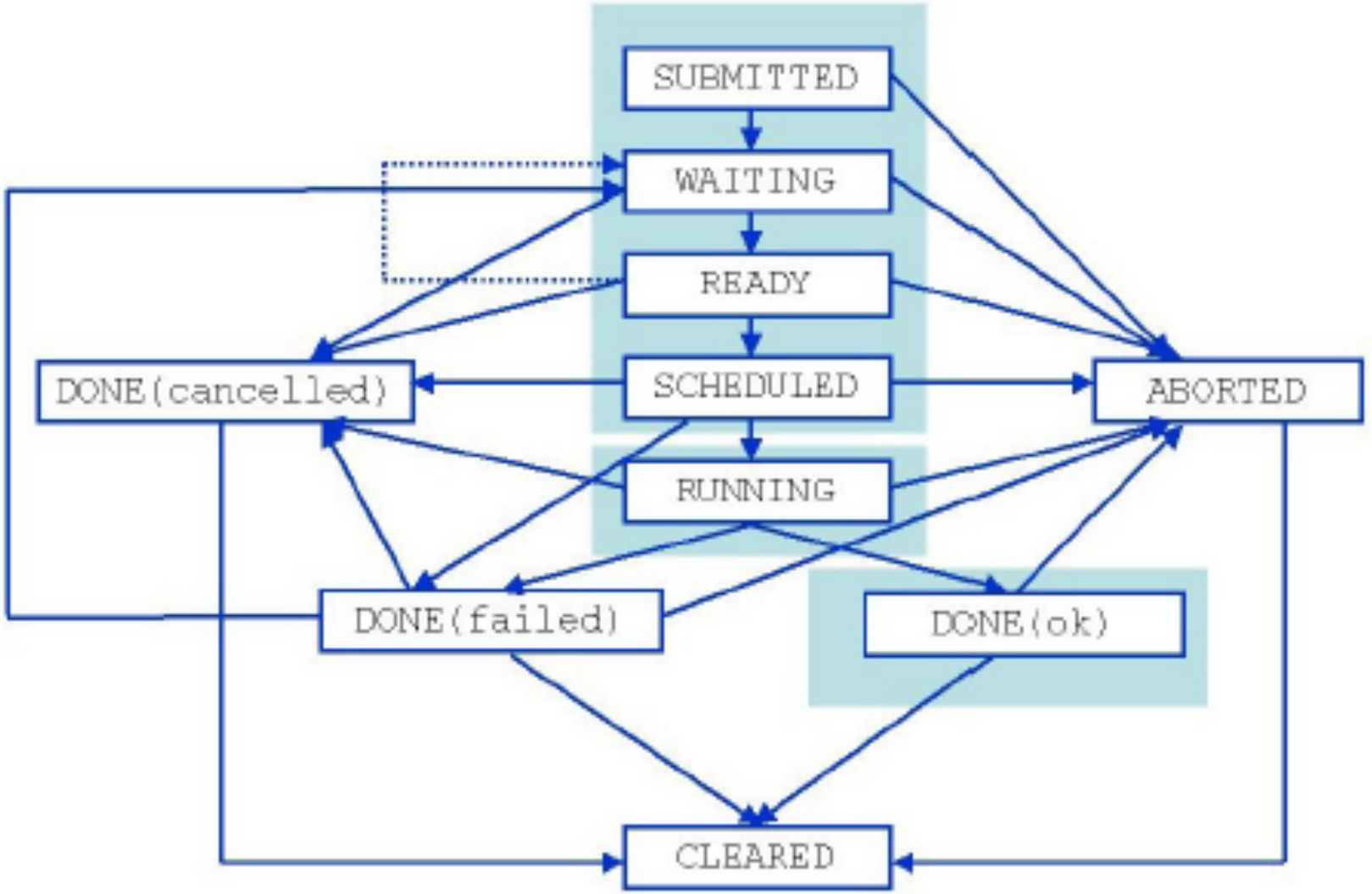


Jobs State Machine (4/9)

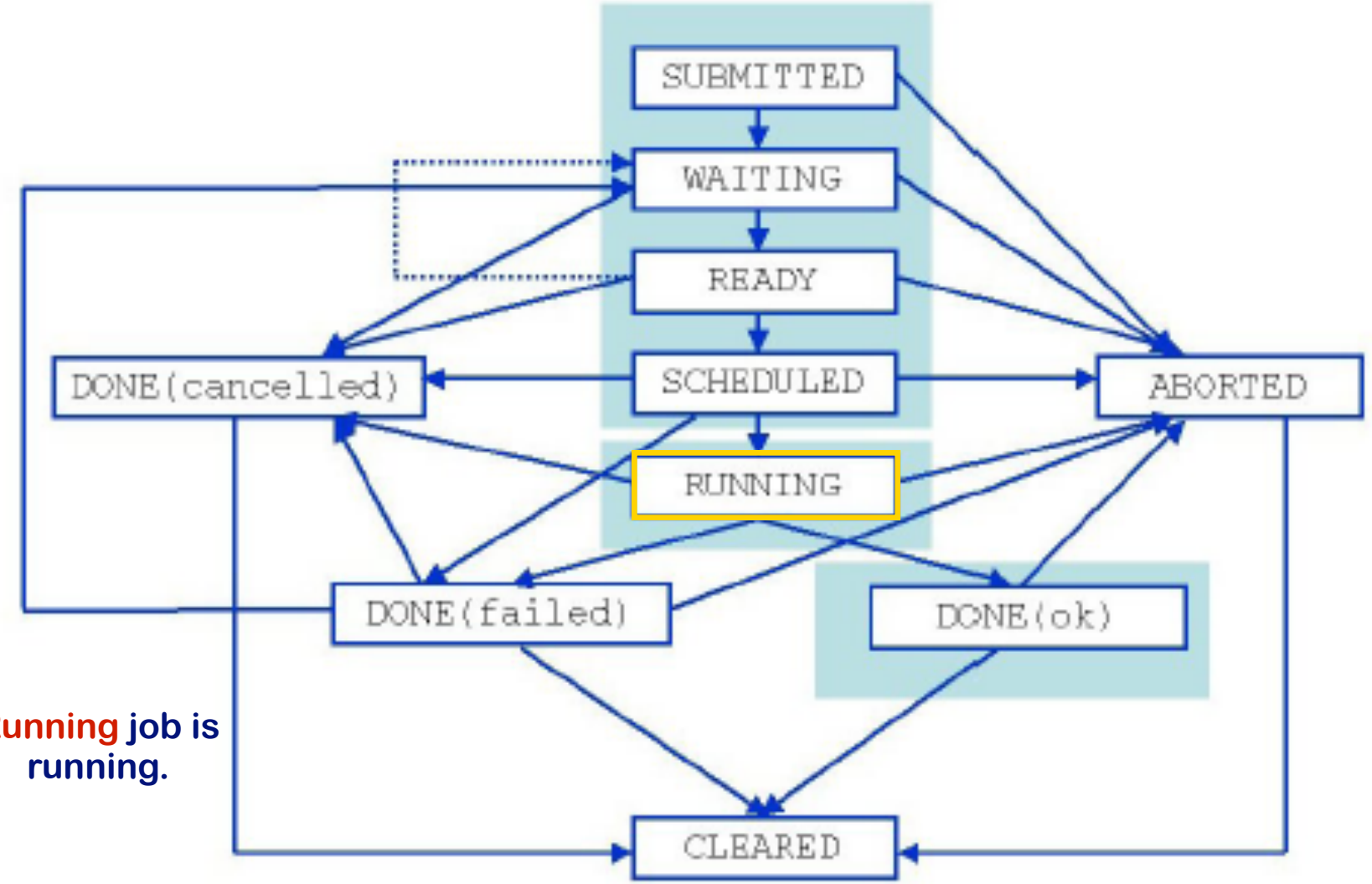


Scheduled job waiting in the queue on the CE.

Jobs State Machine (5/9)

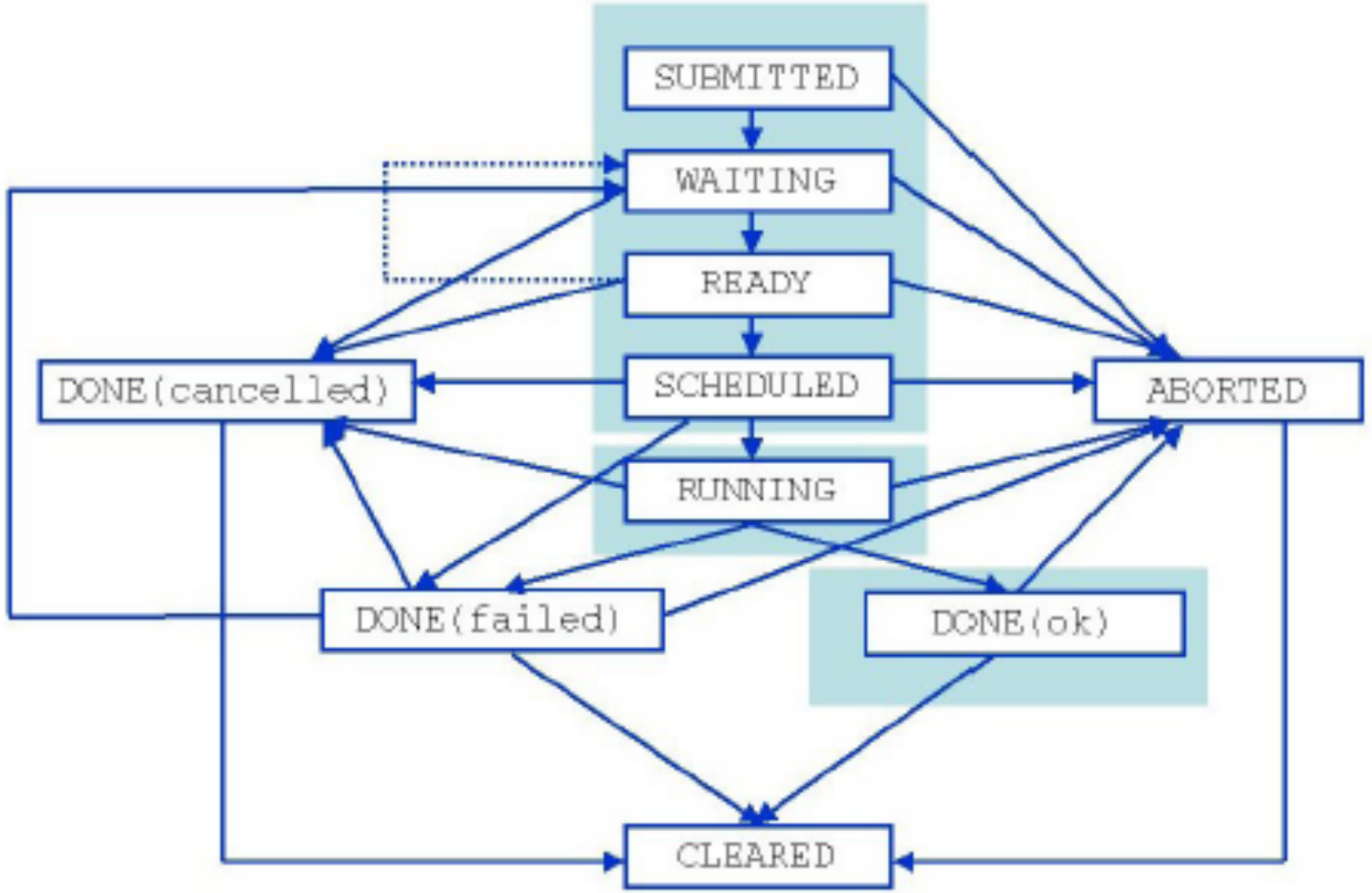


Jobs State Machine (5/9)



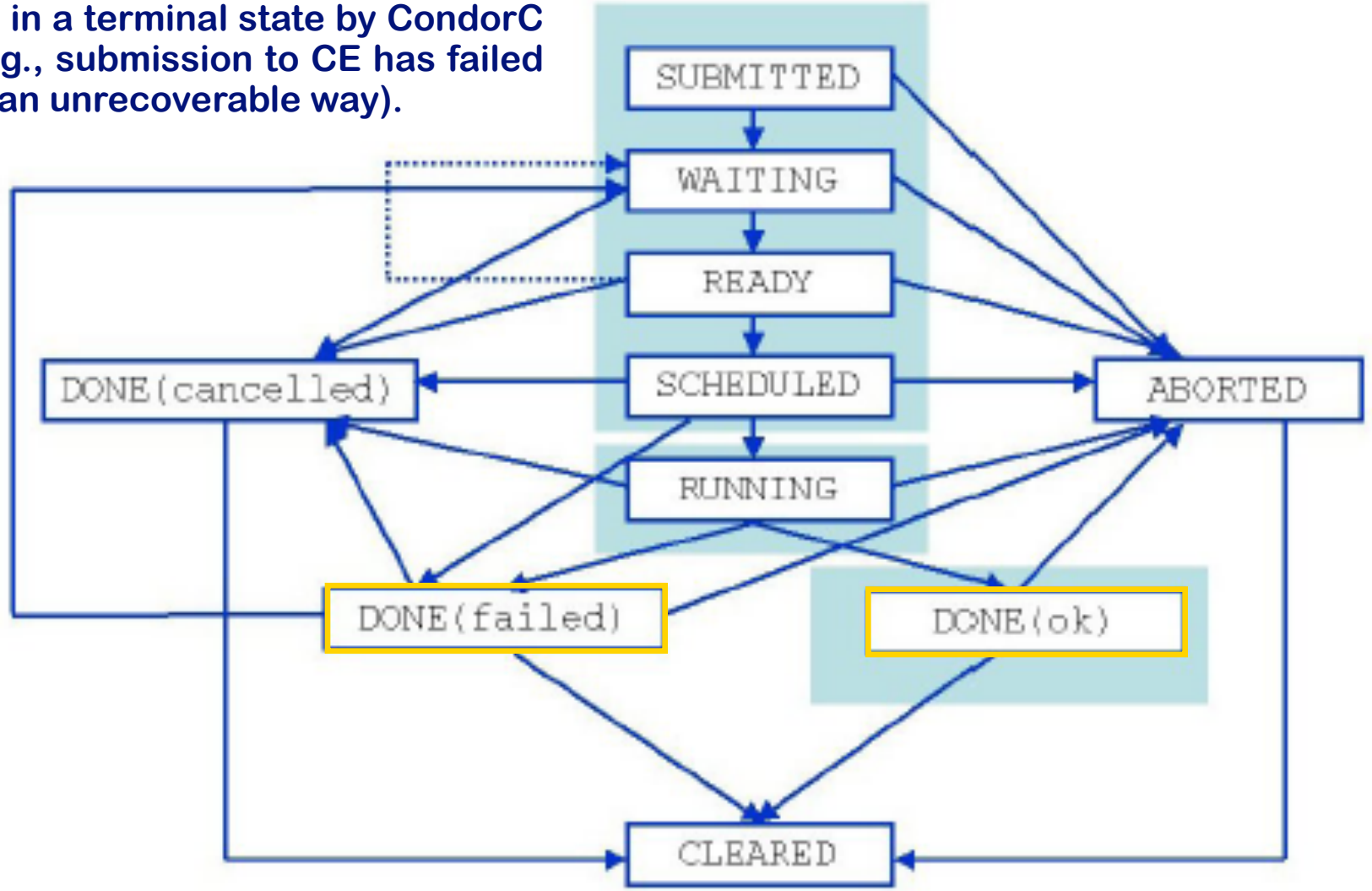
Running job is running.

Jobs State Machine (6/9)

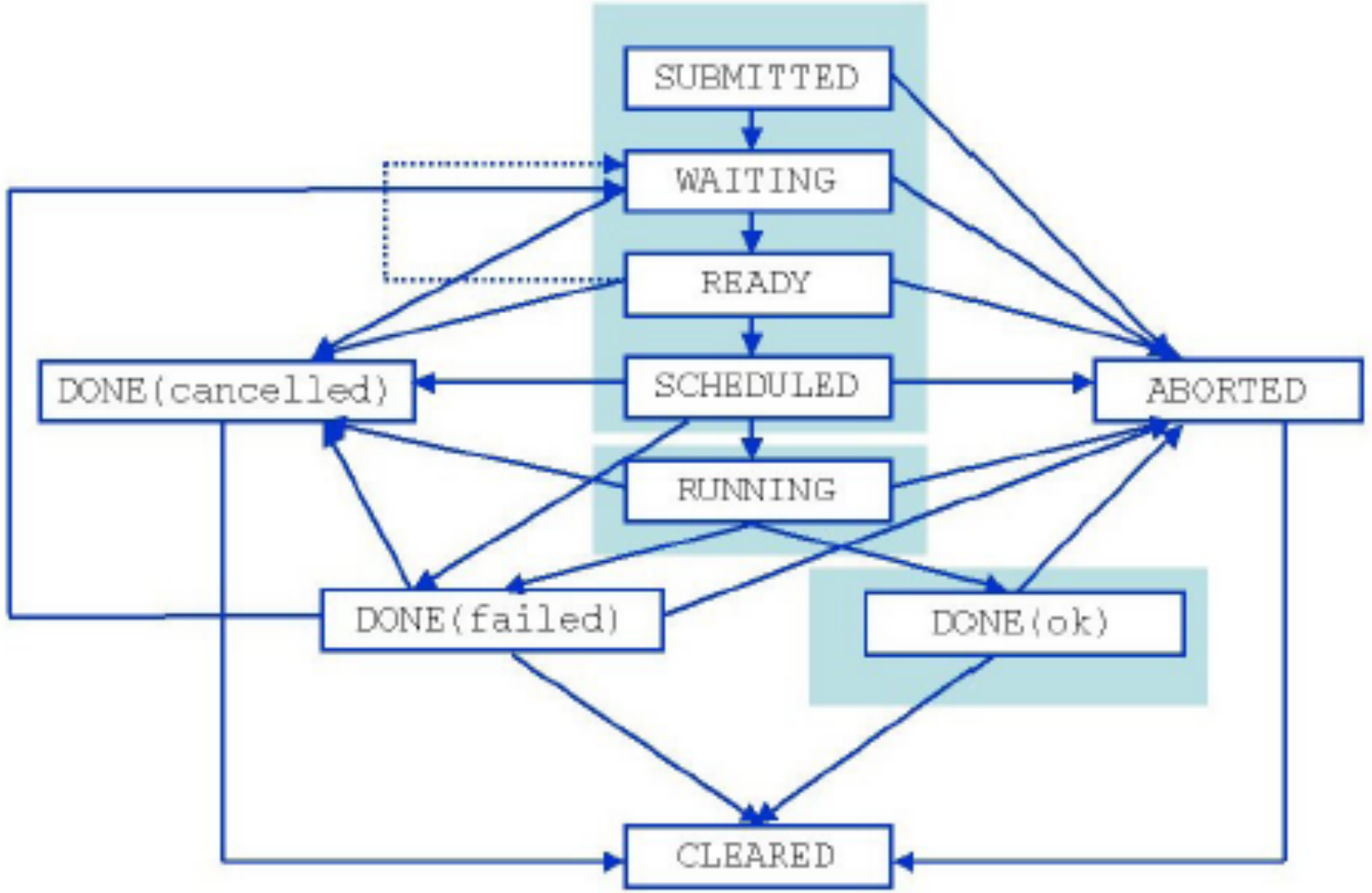


Jobs State Machine (6/9)

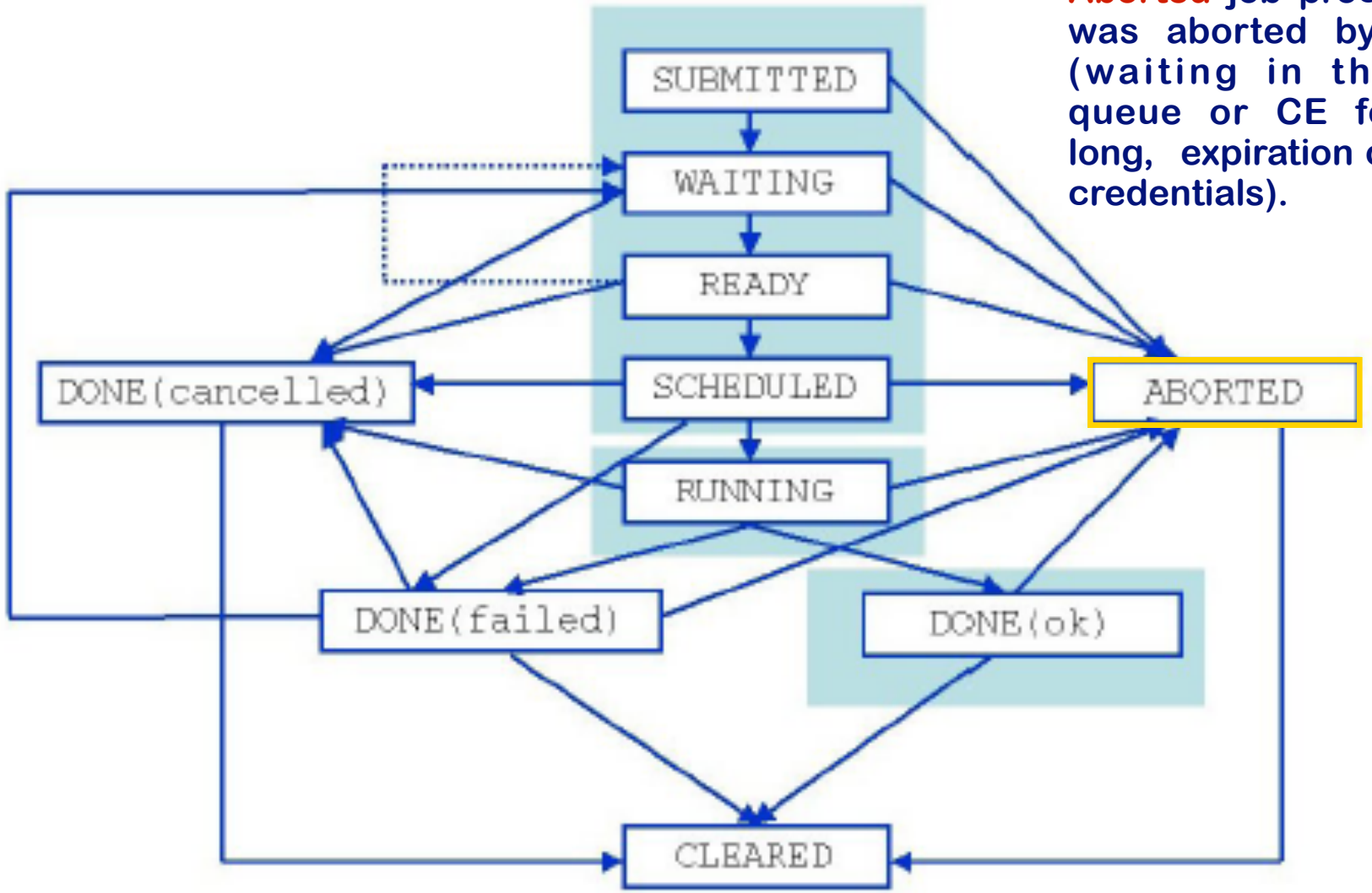
Done job exited or considered to be in a terminal state by CondorC (e.g., submission to CE has failed in an unrecoverable way).



Jobs State Machine (7/9)

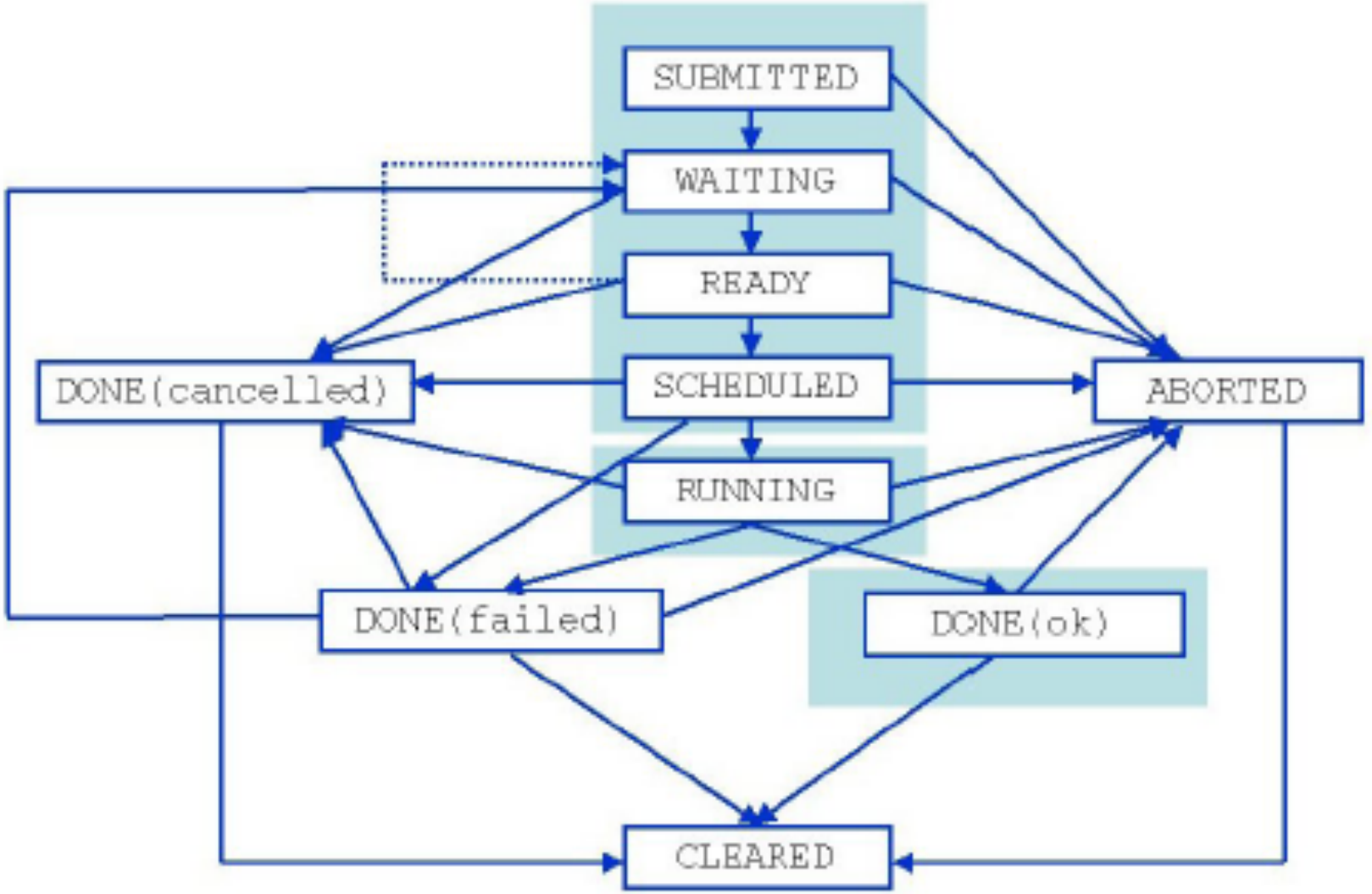


Jobs State Machine (7/9)



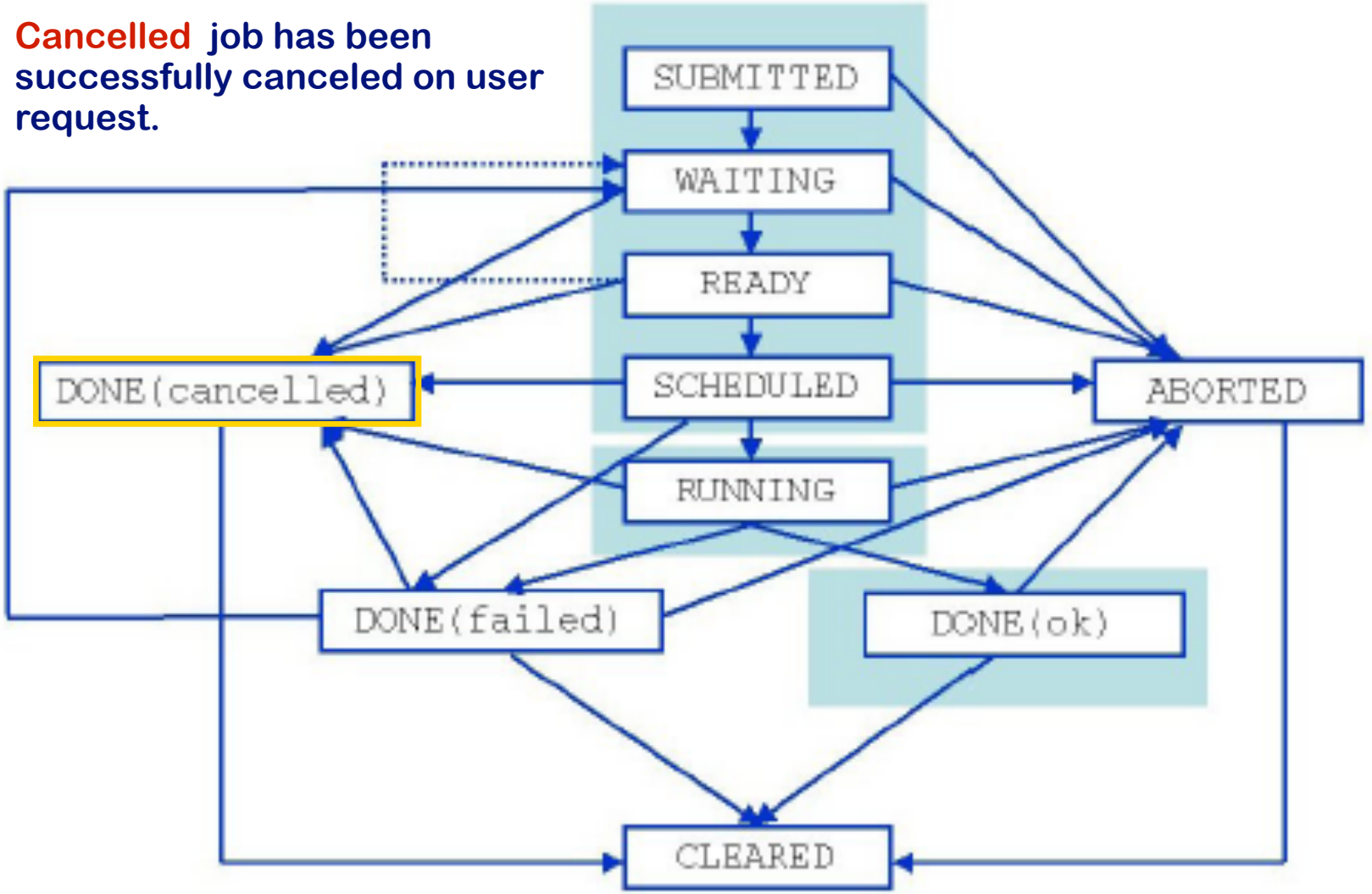
Aborted job processing was aborted by WMS (waiting in the WM queue or CE for too long, expiration of user credentials).

Jobs State Machine (8/9)

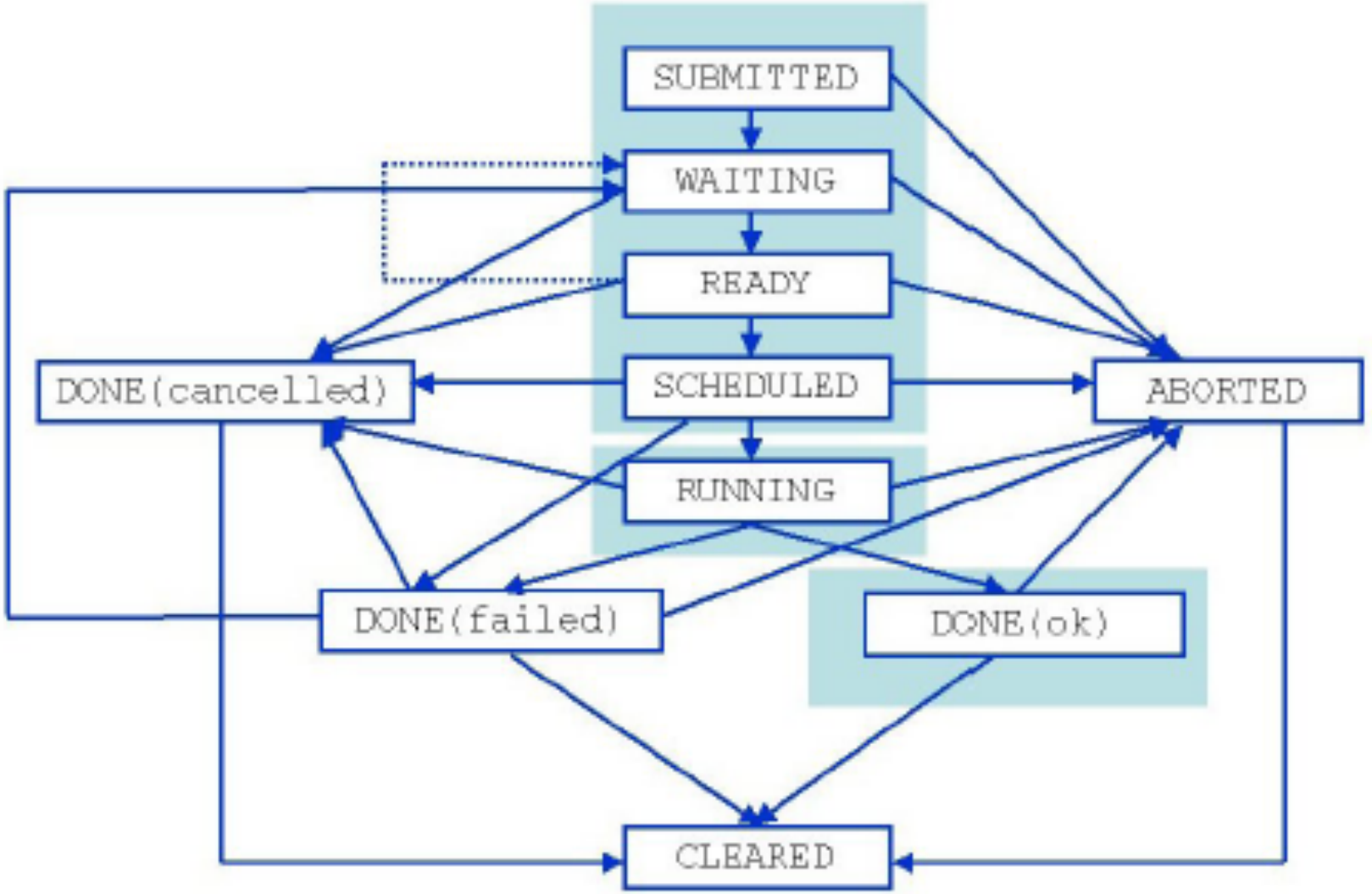


Jobs State Machine (8/9)

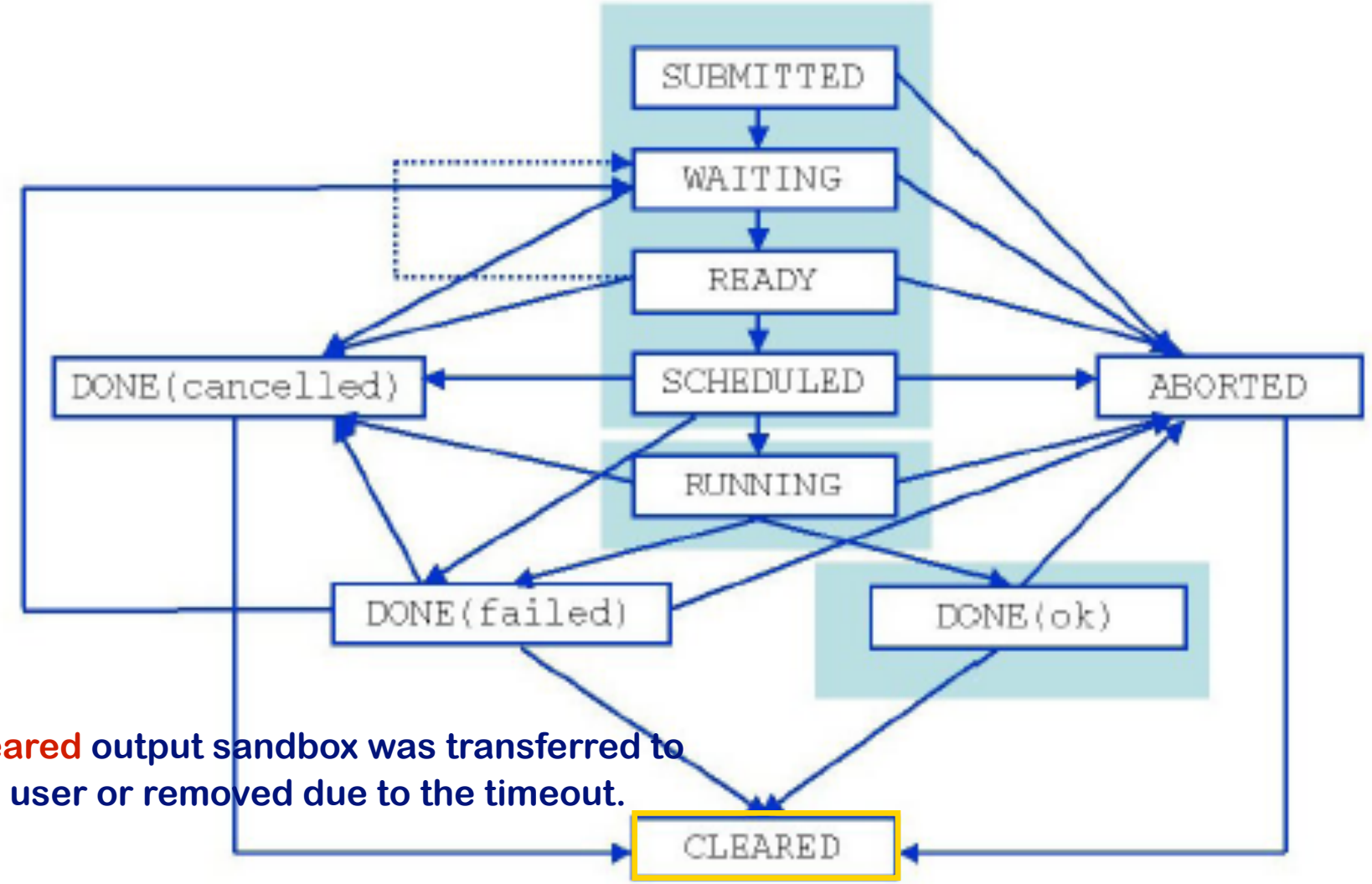
Cancelled job has been successfully canceled on user request.



Jobs State Machine (9/9)



Jobs State Machine (9/9)



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

References

WMProxy User's guide

- https://edms.cern.ch/file/674643/1/WMPROXY-guide.pdf

JDL Attributes Specification

- https://edms.cern.ch/file/590869/1/EGEE-JRA1-TEC-590869-JDL-Attributes-v0-9.pdf

gLite User's guide

- https://edms.cern.ch/file/722398/1.2/gLite-3-UserGuide.pdf

Questions …