

# Architecture of the WMS



Dr. Giuliano Taffoni

I N A F



INFORMATION  
SYSTEMS UNIT



# Outline

**This presentation will cover the following arguments:**

- **Overview of WMS Architecture**
- **Job Description Language Overview**
- **WMProxy overview**

 **= New in gLite 3.0**

# First Part

## Architecture of the gLite WMS



Scientific Instruments and sensors in the Grid, ICTP Trieste, 24.04.2007

# Workload Manager Services



# Workload Manager Services

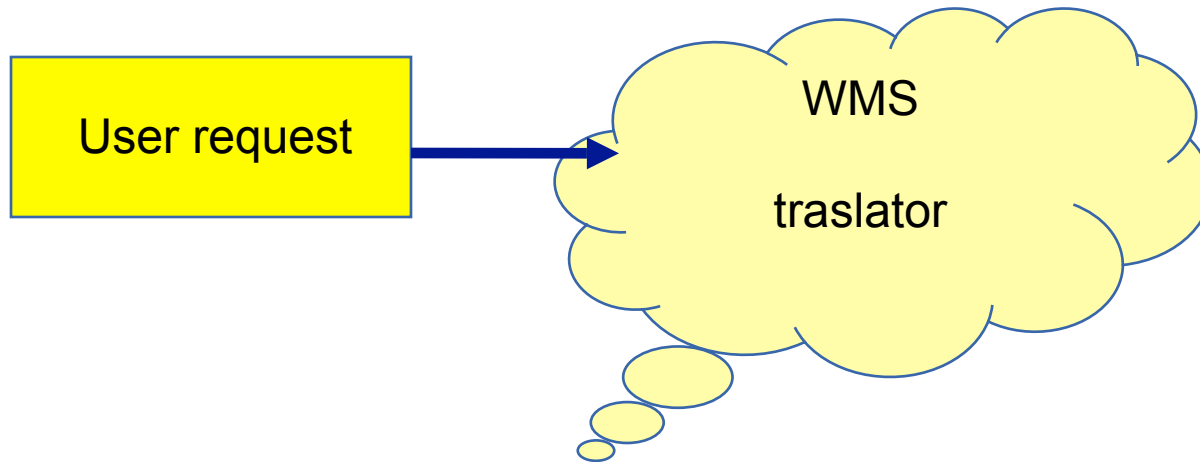
User request

# Workload Manager Services

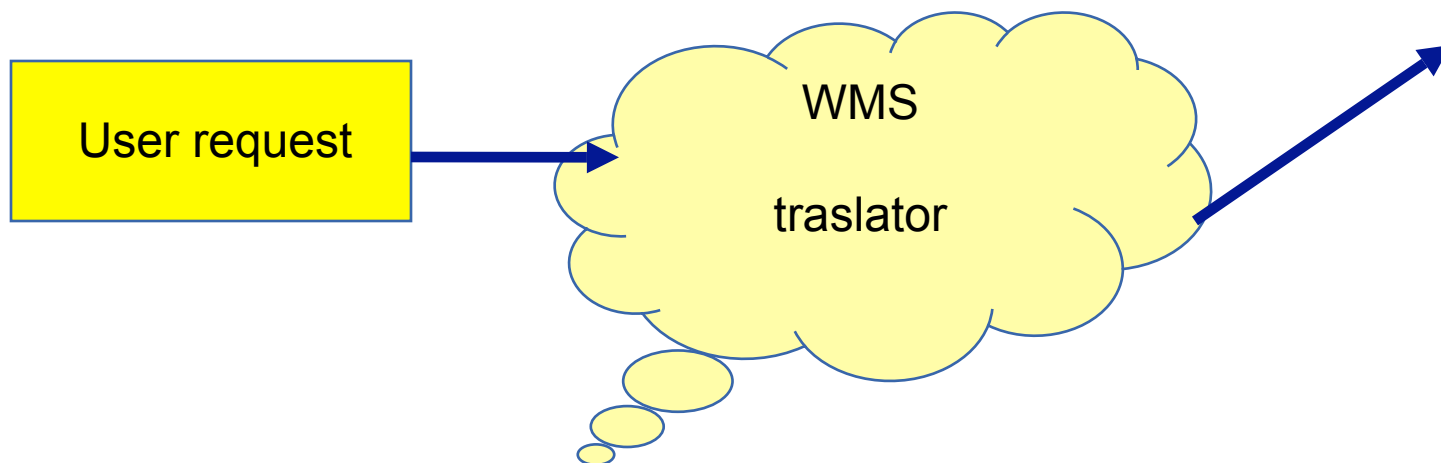
User request



# Workload Manager Services

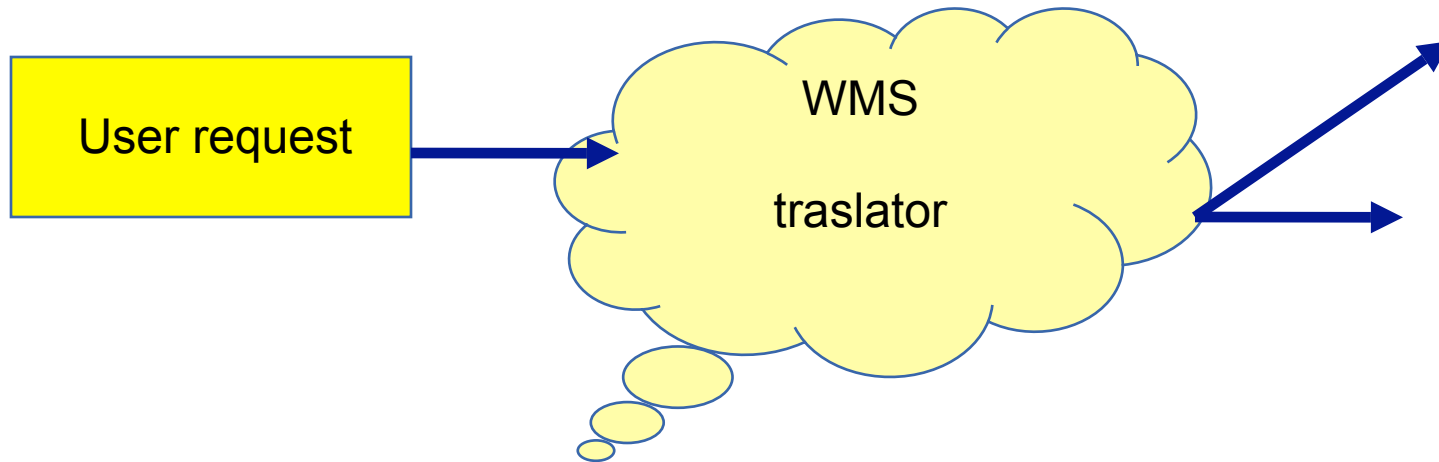


# Workload Manager Services

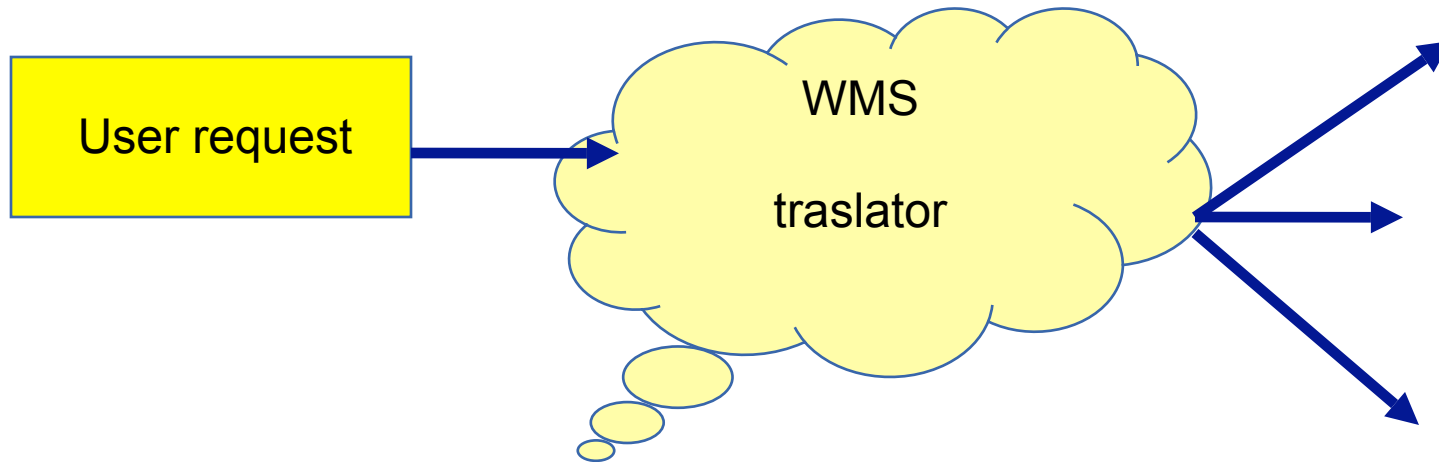




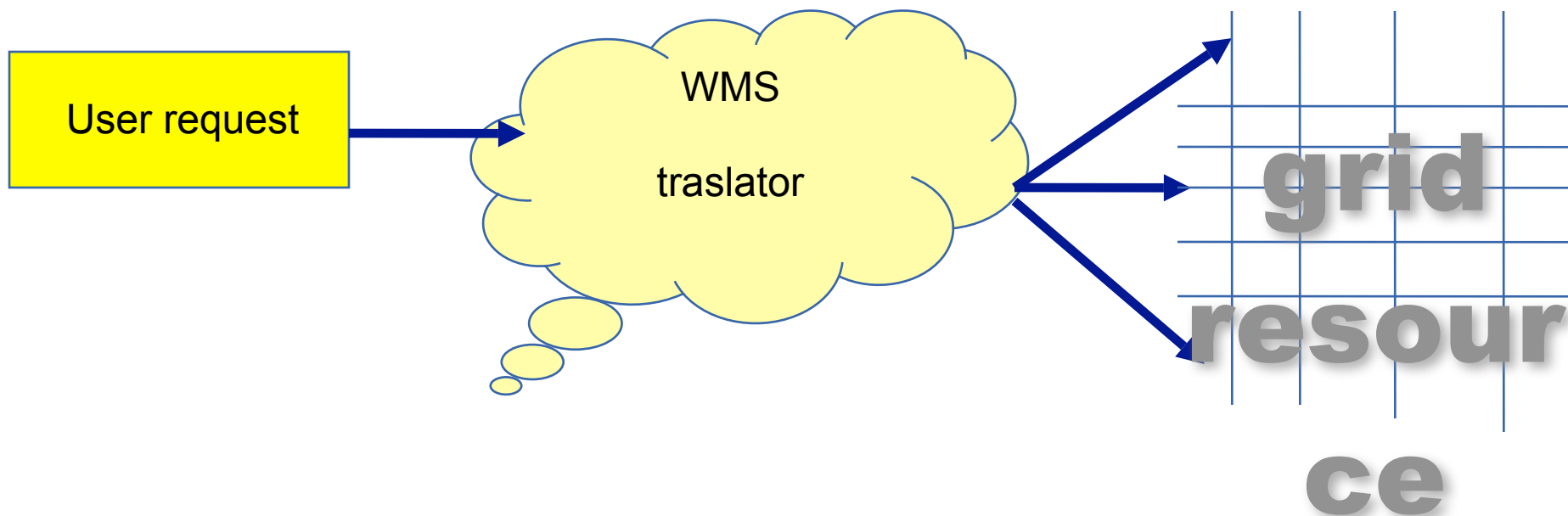
# Workload Manager Services



# Workload Manager Services



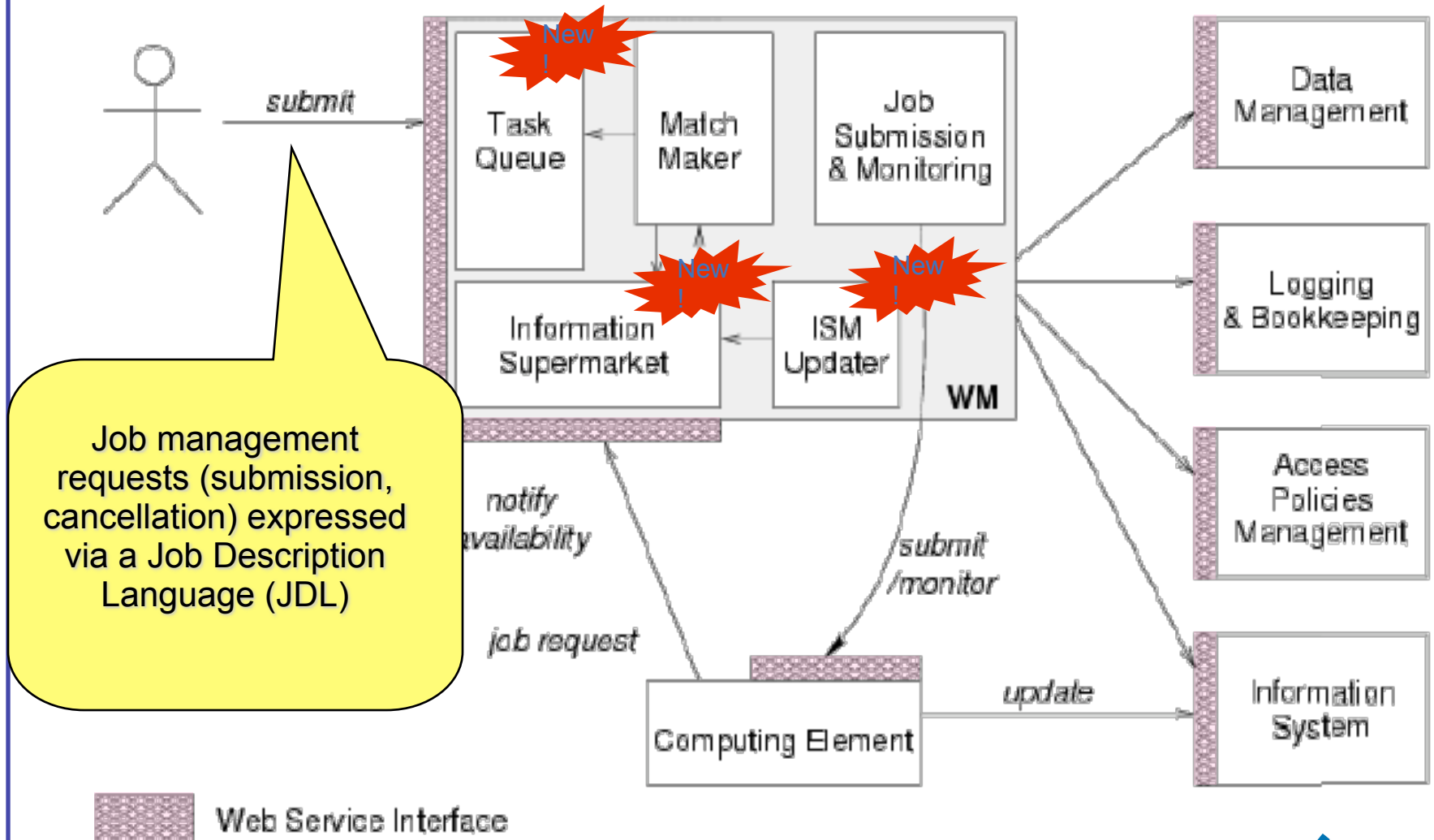
# Workload Manager Services



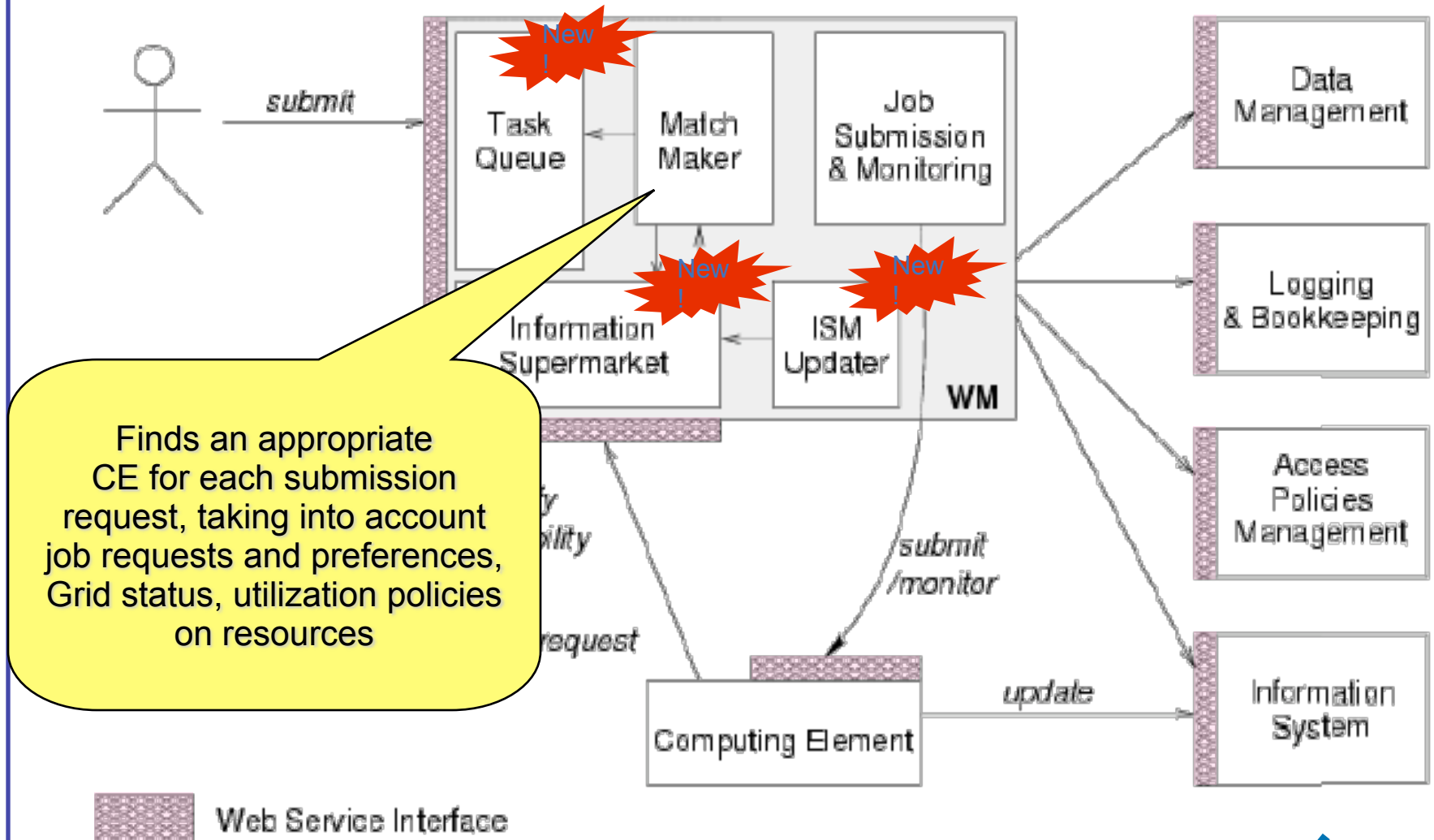
# WMS Objectives

- 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 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

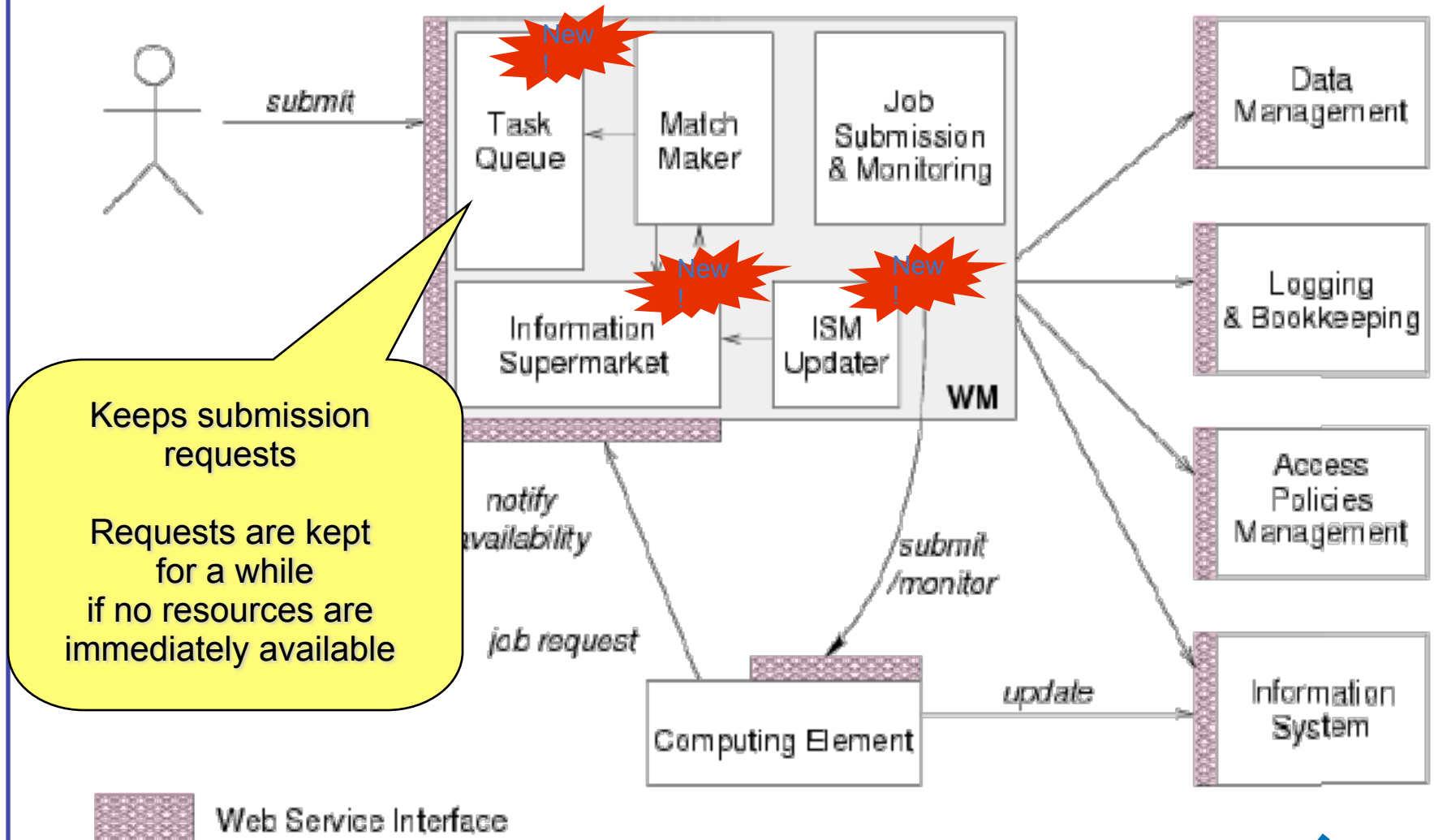
# WMS Architecture



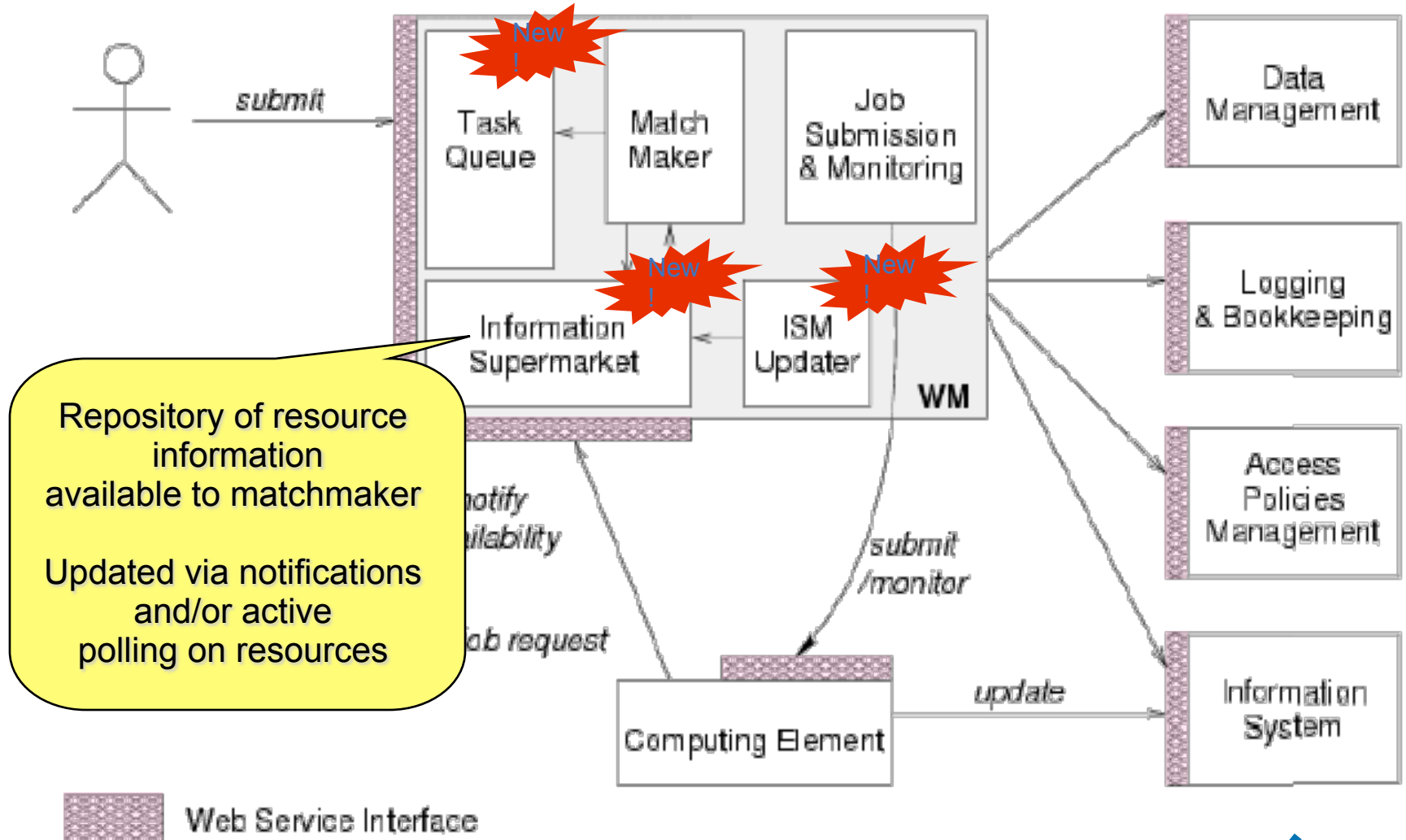
# WMS Architecture



# WMS Architecture

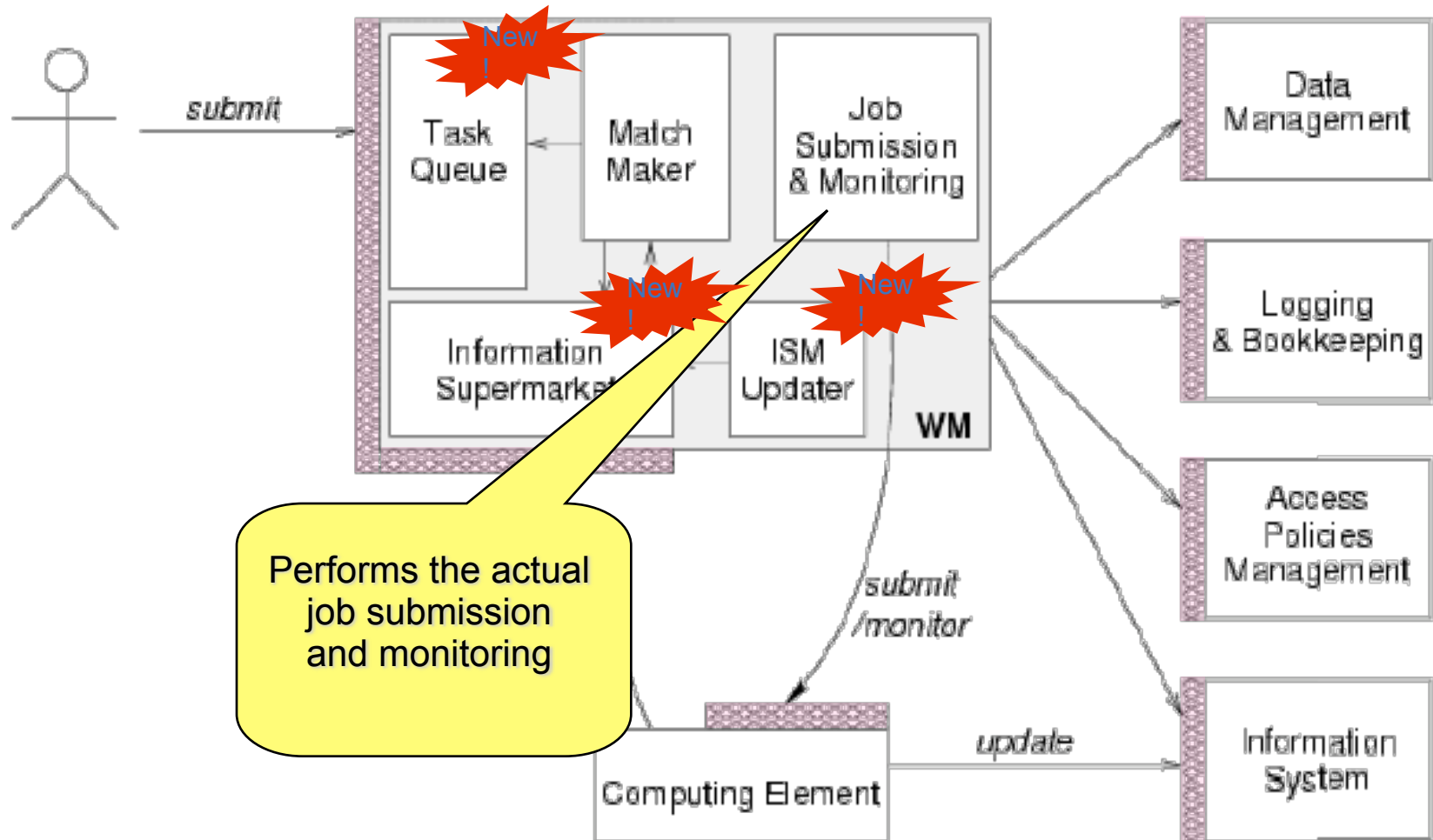


# WMS Architecture





# WMS Architecture




Web Service Interface

# NS and WMPProxy


- The **Network Server (NS)** is a generic network daemon that provides support for the job control functionality. It is responsible for accepting incoming requests from the WMS-UI (e.g. job submission, job removal), which, if valid, are then passed to the Workload Manager.
- The **Workload Manager Proxy (WMPProxy)** is a service providing access to WMS functionality through a Web Services based interface. Besides being



# WMS Information Supermarket

- ISM represents one of the most notable improvements  in the WM
- The ISM basically consists of a repository of resource information that is available in read only mode to the matchmaking engine
  - the update is the result of

# WMS Task Queue

- The Task Queue represents the second most notable improvement in the **WMS** internal design 
  - possibility to keep a submission request for a while if no resources are immediately available that match the job requirements
    - technique used by the AliEn and Condor systems
- Non-matching requests
  - will be retried either periodically
    - eager scheduling approach

# WMS Job Submission Services

WMS components handling the job during its lifetime and performs the submission

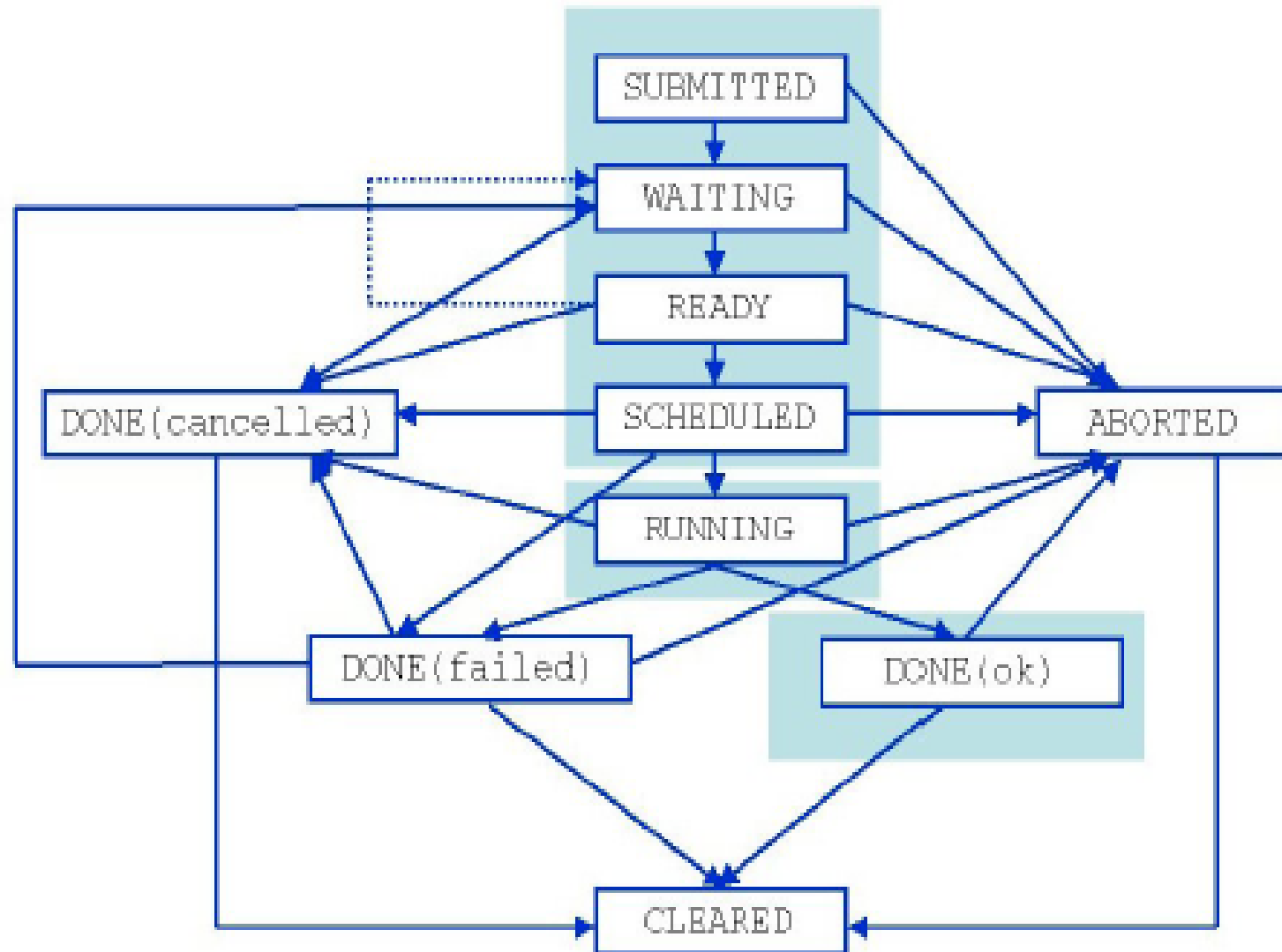
- **Job Adapter (JA)**
  - is responsible for
    - making the final touches to the JDL expression for a job, before it is passed to CondorC for the actual submission
    - creating the job wrapper script that creates the appropriate execution environment in the CE worker node
      - transfer of the input and of the output sandboxes
- **CondorC**
  - responsible for
    - performing the actual job management operations
      - job submission, job removal
- **DAGMan**
  - meta-scheduler
    - purpose is to navigate the graph
    - determine which nodes are free of dependencies
    - follow the execution of the corresponding jobs



# WMS Job Submission Services

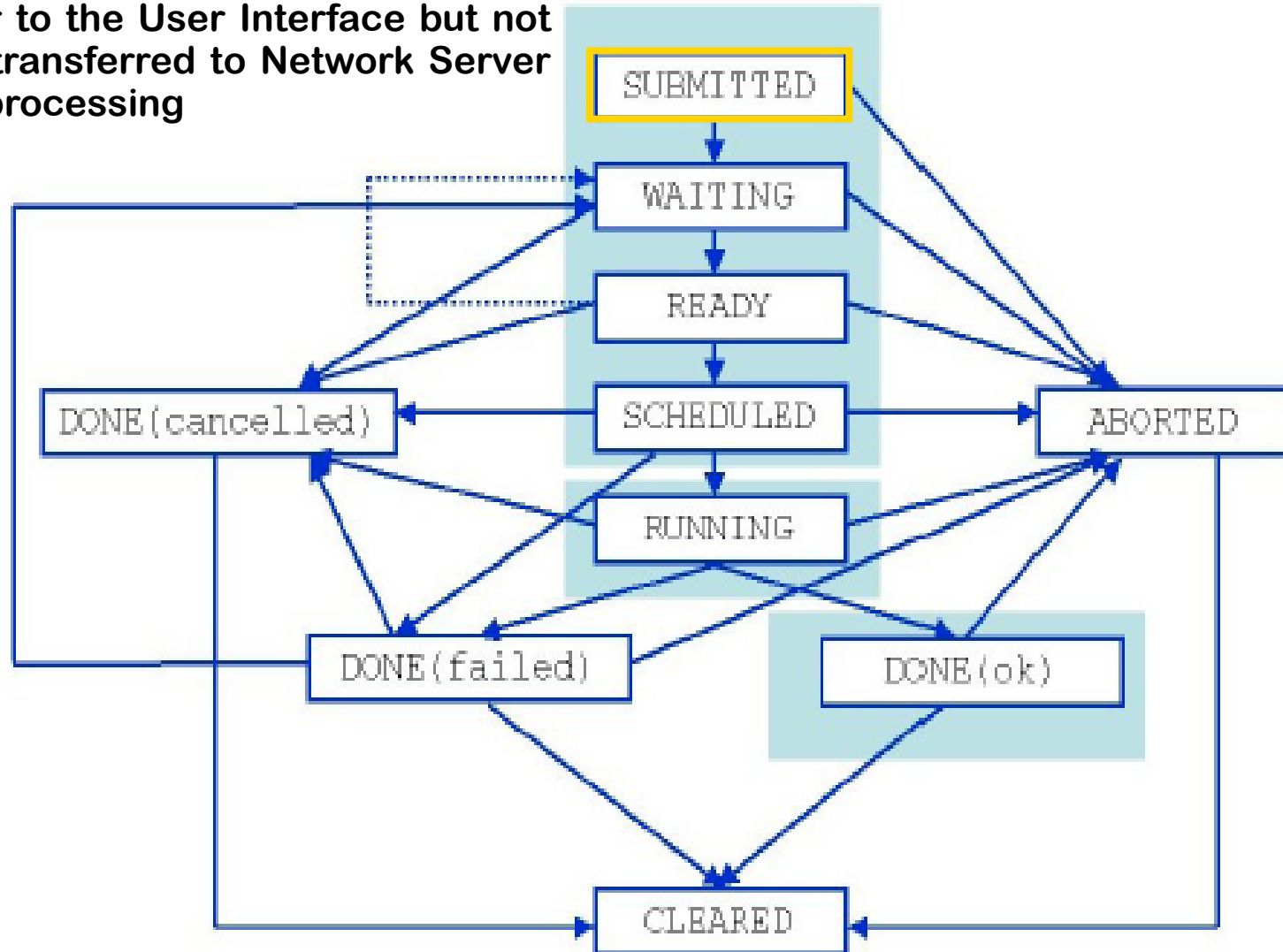
- **Log Monitor (LM)**
  - is responsible for
    - watching the CondorC log file
    - intercepting interesting events concerning active jobs
- **Proxy Renewal Service**
  - is responsible for assuring that,
    - for all the lifetime of a job, a valid user proxy exists within the WMS
    - MyProxy Server is contacted in order to renew the user's credential
- **Logging & Bookkeeping (LB)**
  - is responsible for
    - Storing events generated by the various components of the WMS
    - Delivering to the user information about the job's status

# WMS Job Submission Services



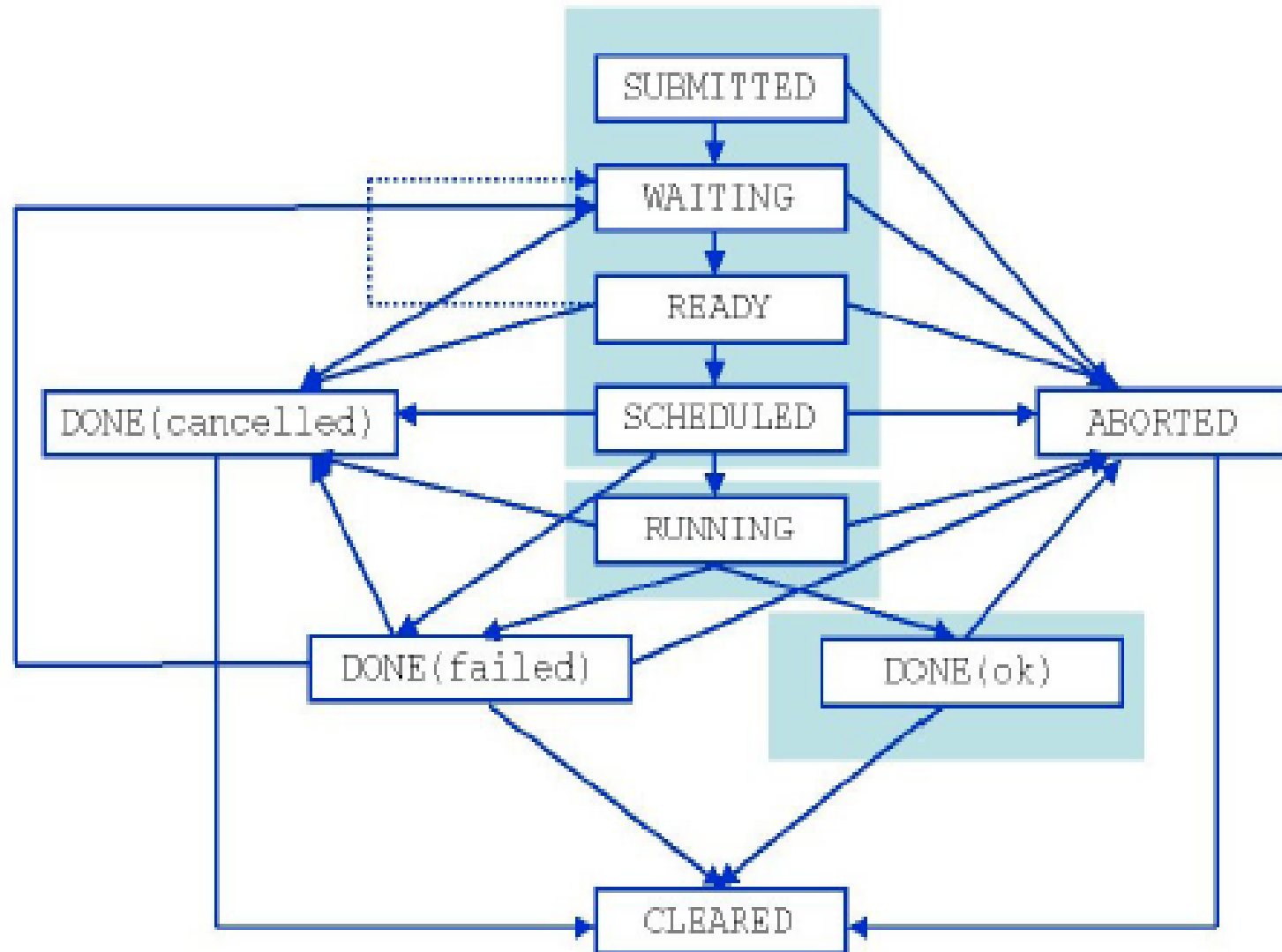
# WMS Job Submission Services

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



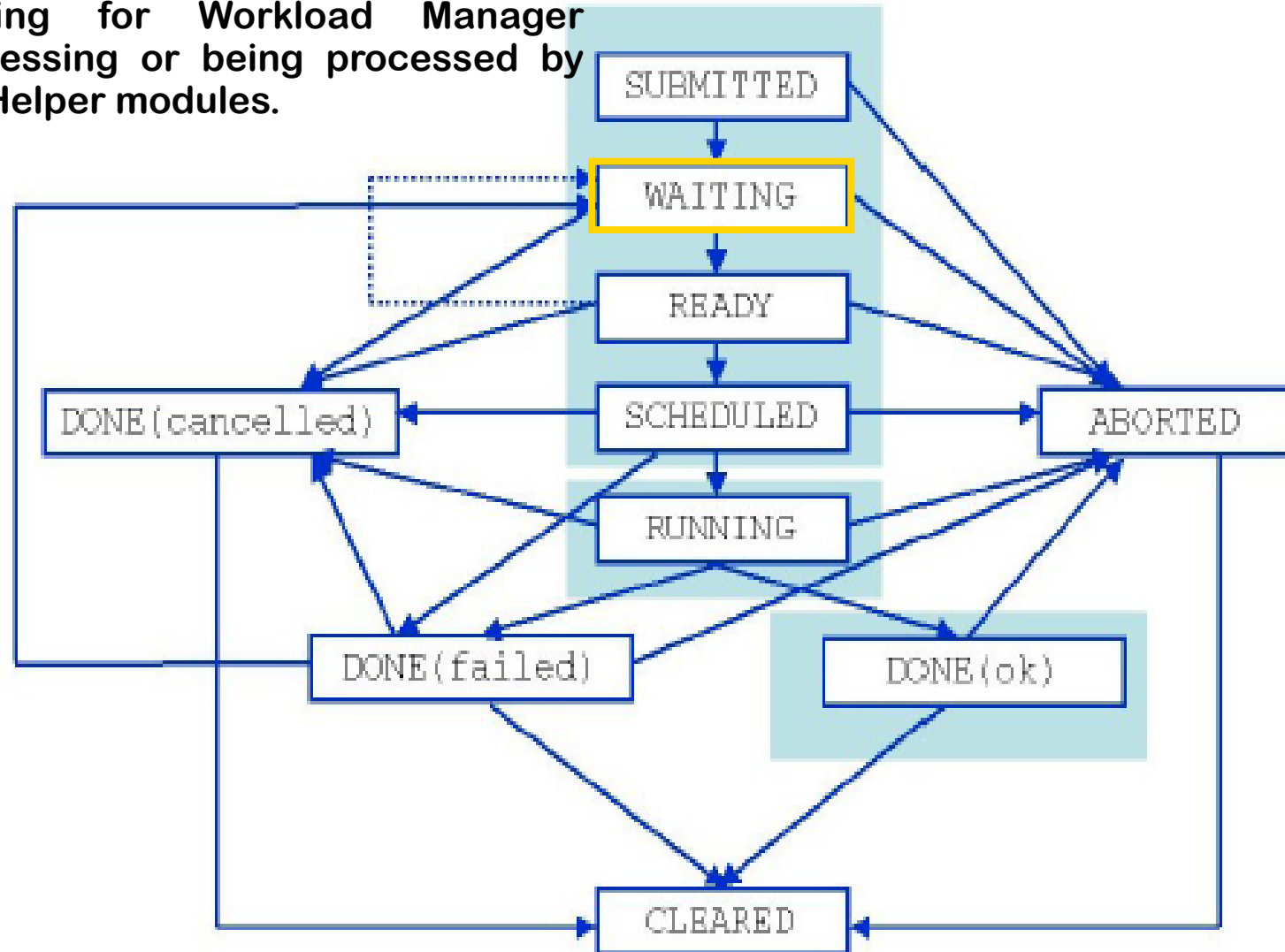


# WMS Job Submission Services

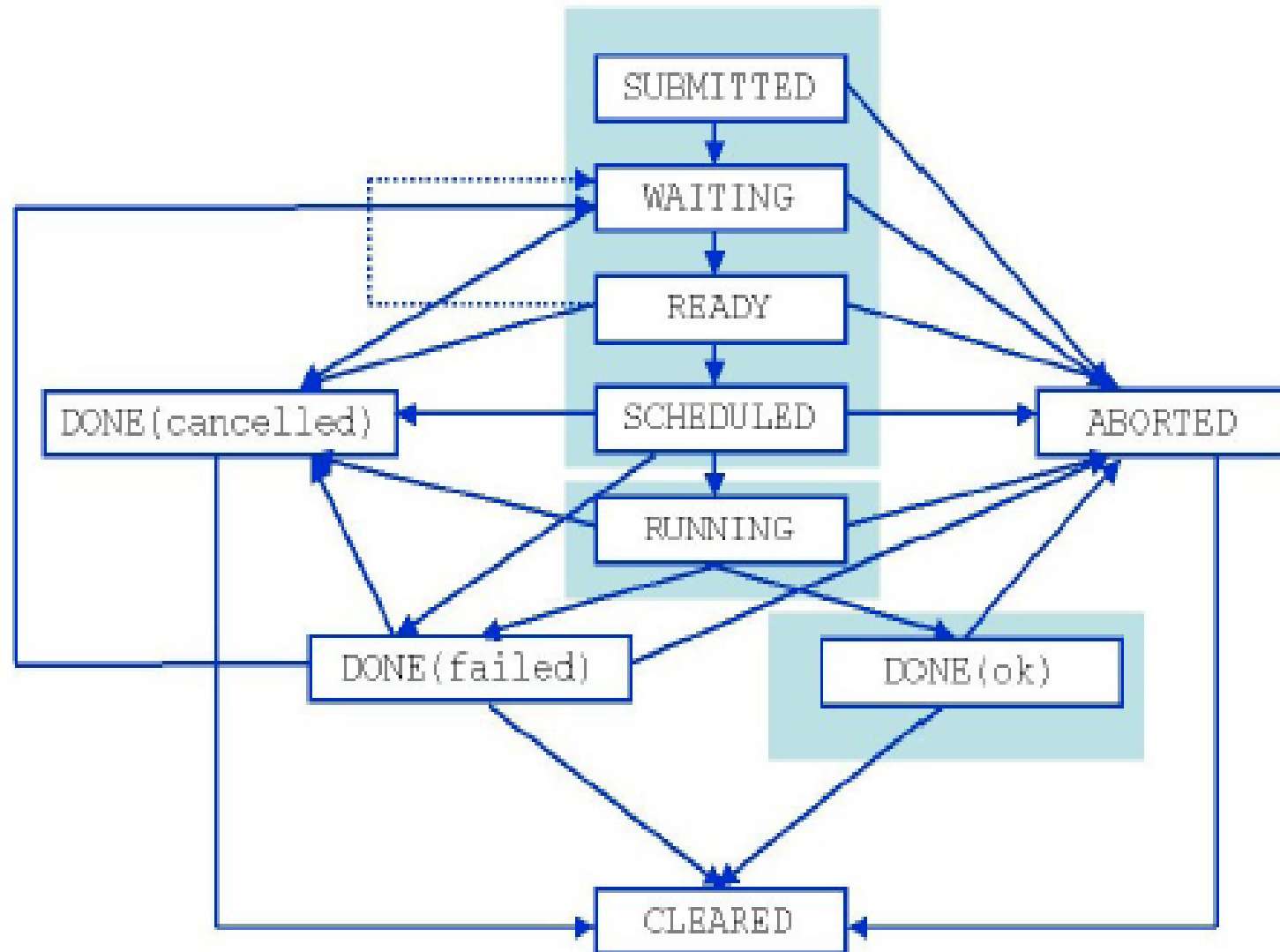


# WMS Job Submission Services

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

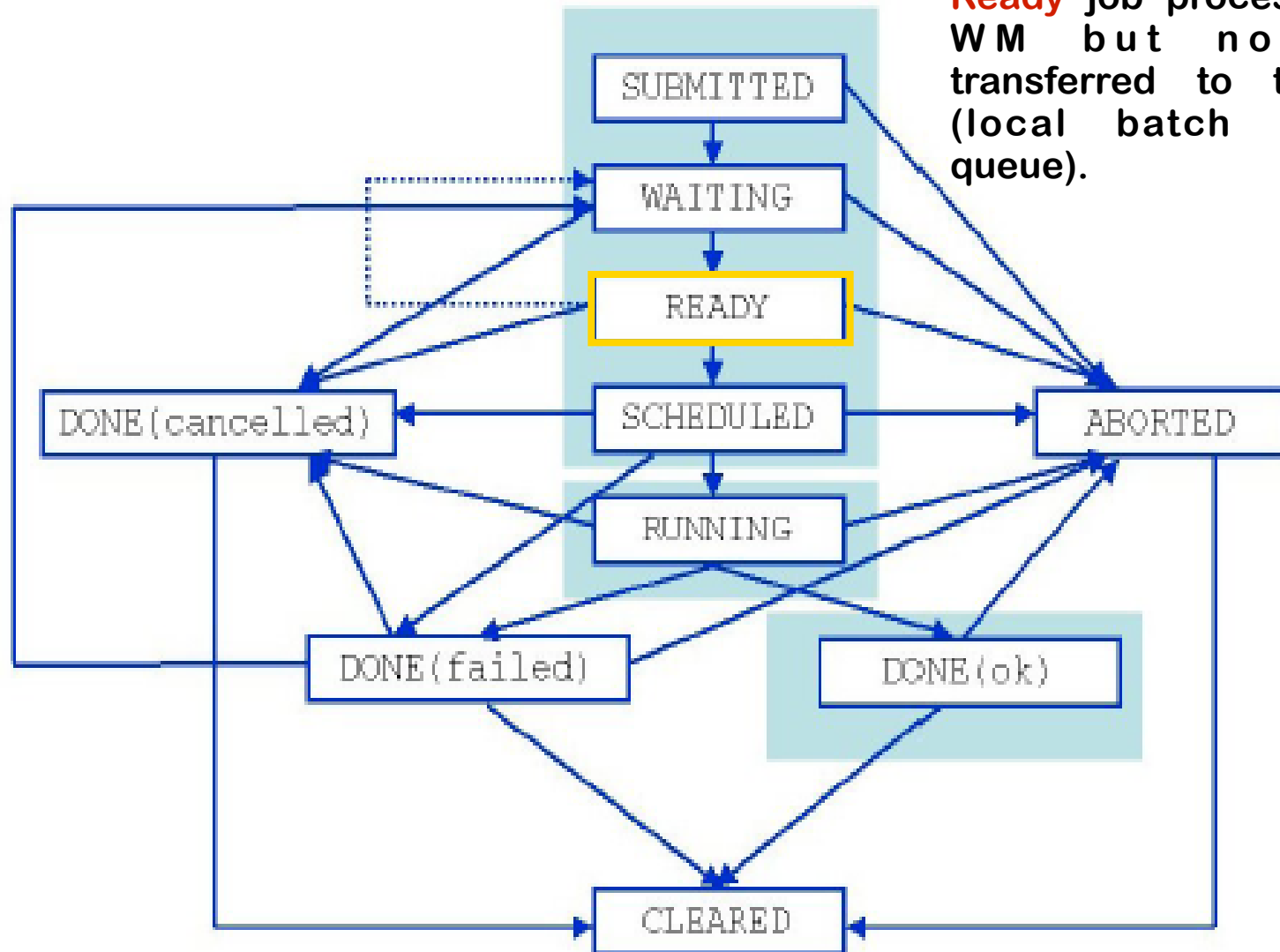


# WMS Job Submission Services

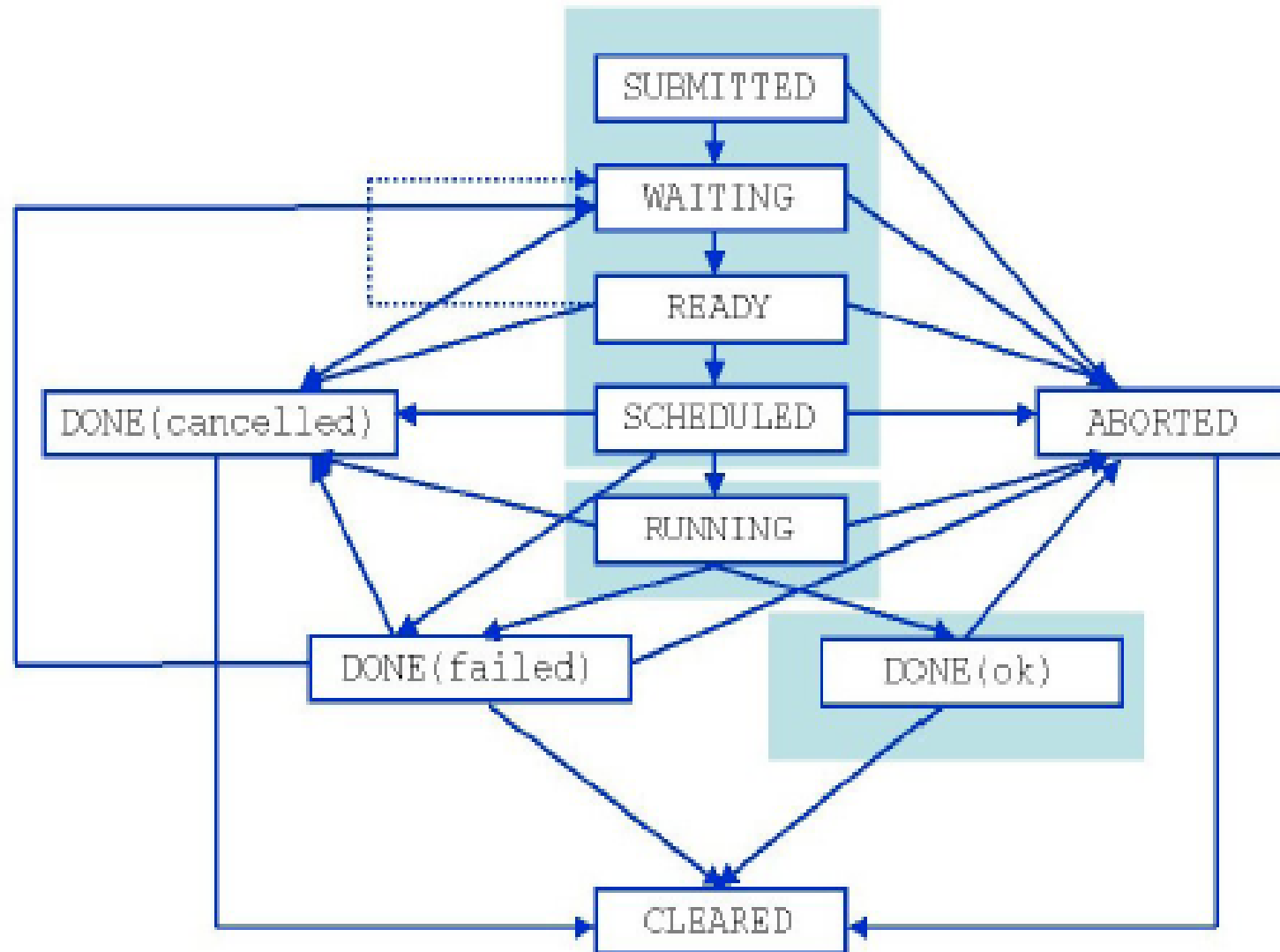


# WMS Job Submission Services

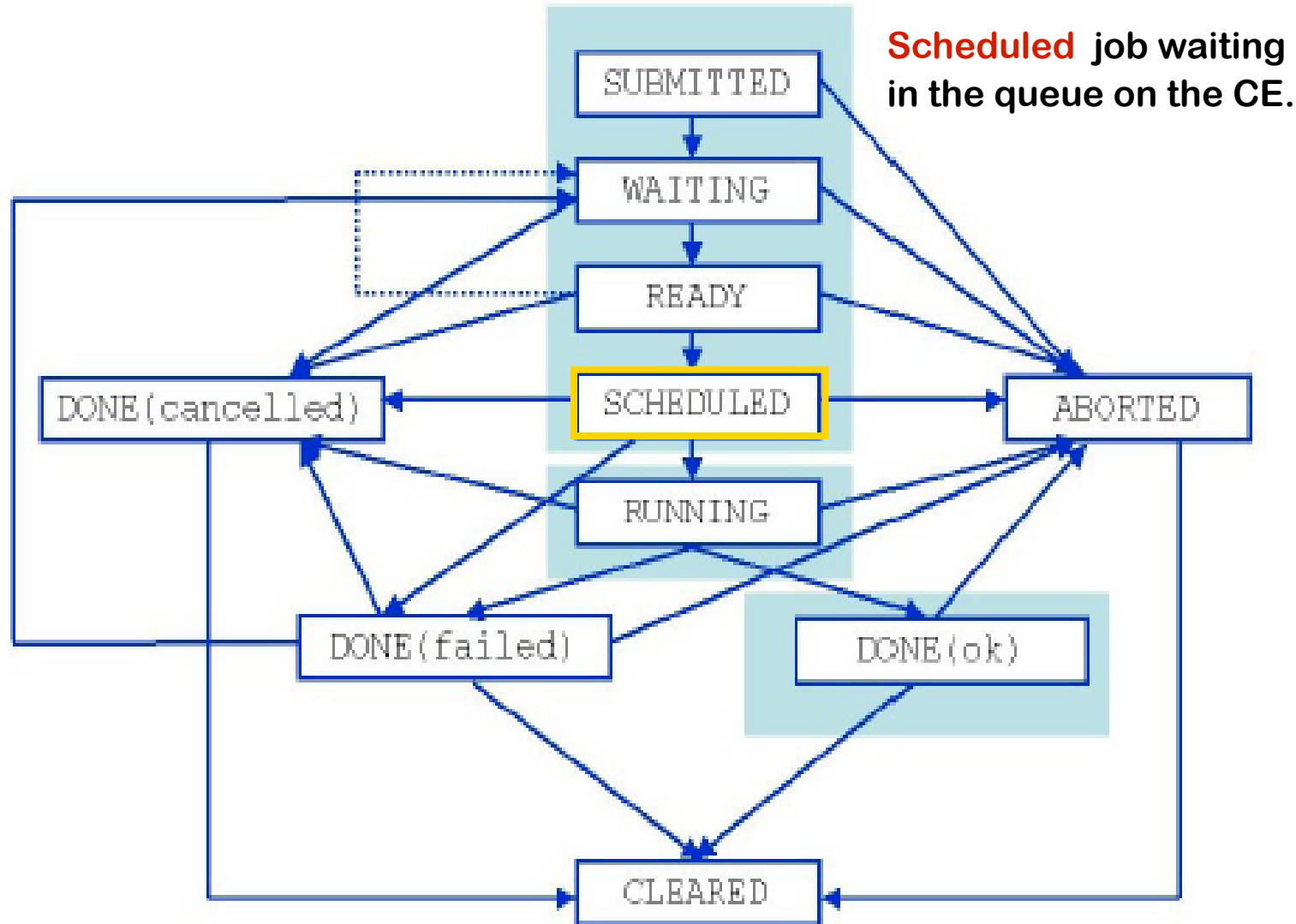
**Ready** job processed by WM but not yet transferred to the CE (local batch system queue).



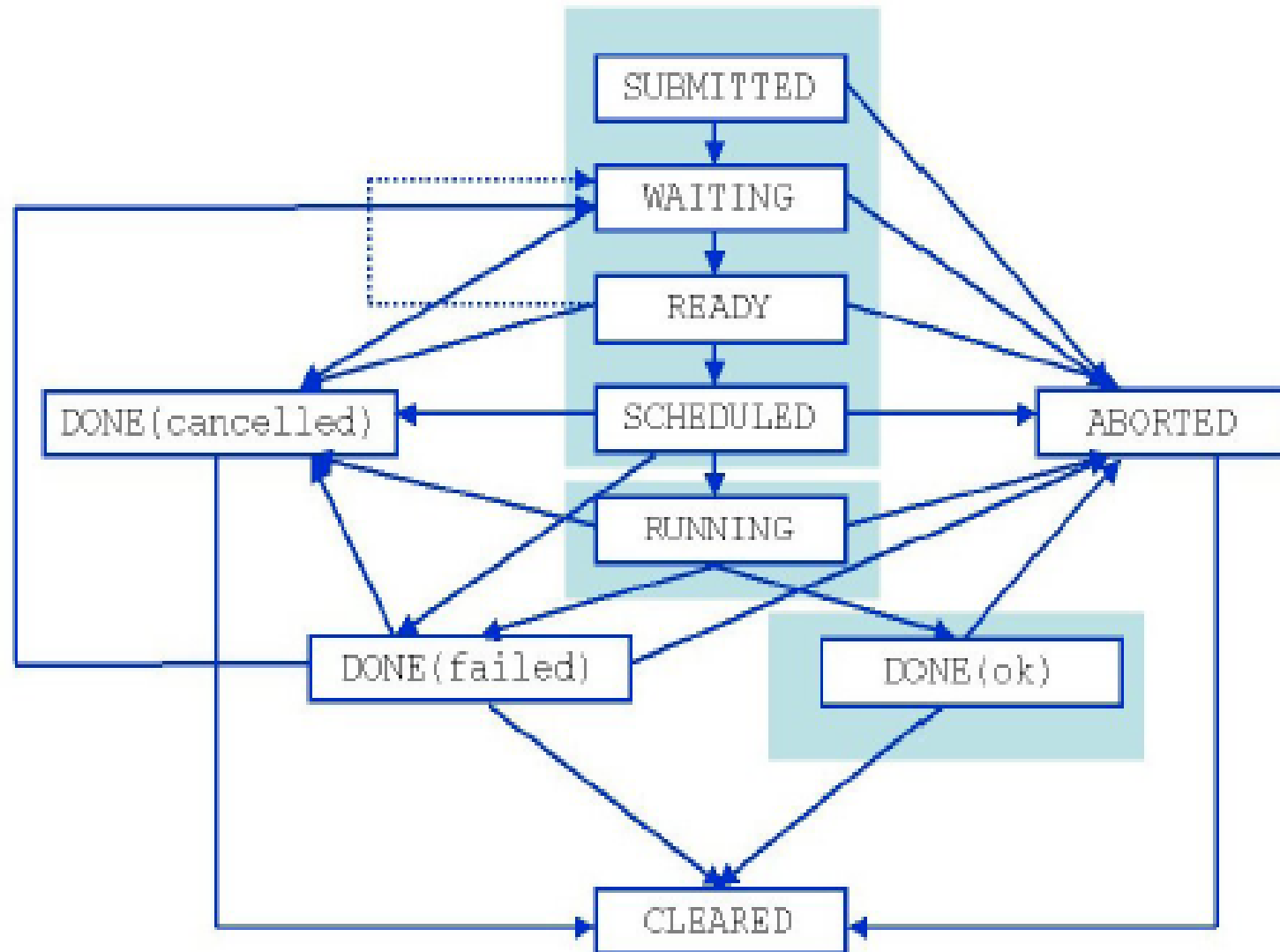
# WMS Job Submission Services



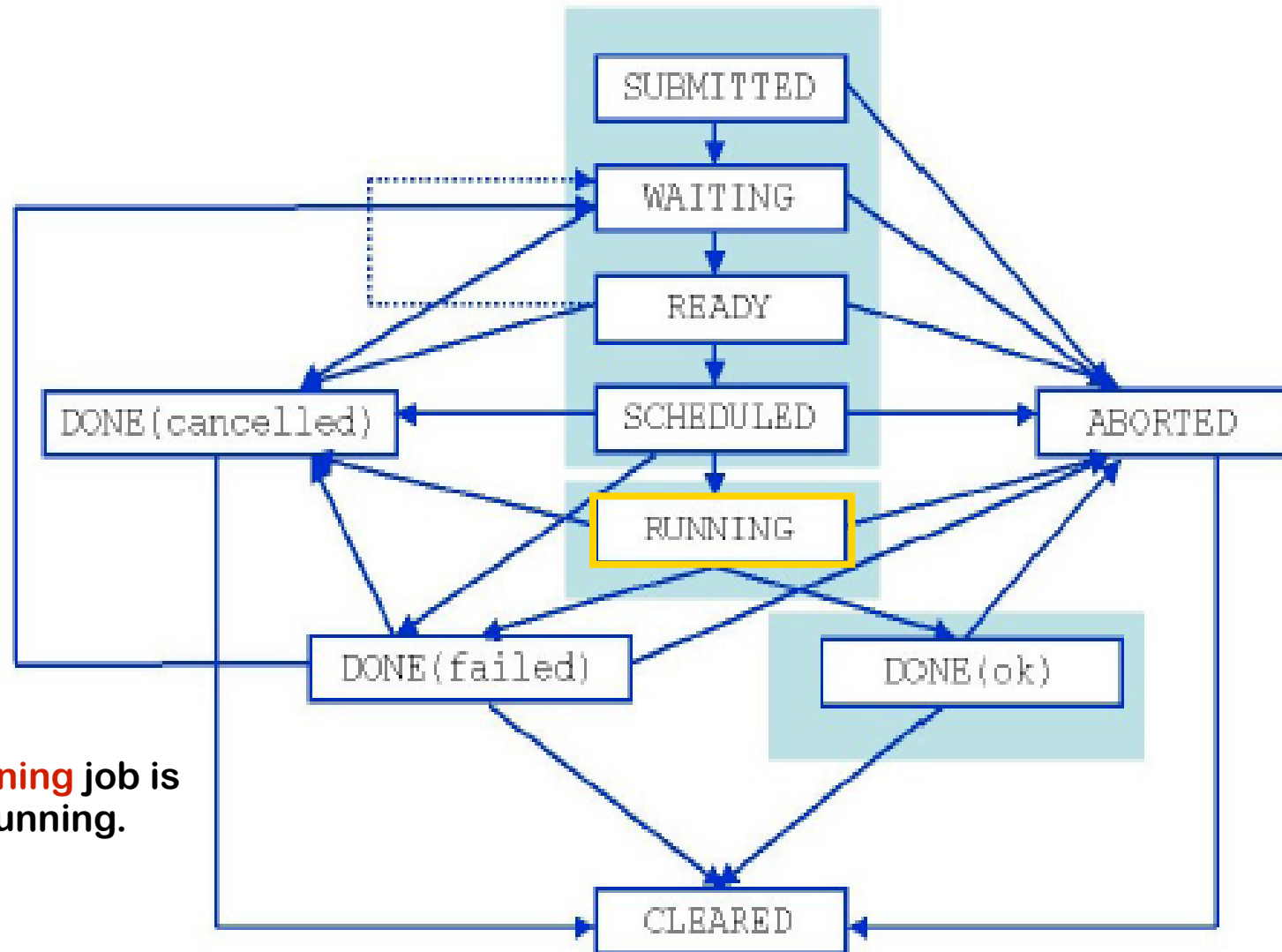
# WMS Job Submission Services



# WMS Job Submission Services



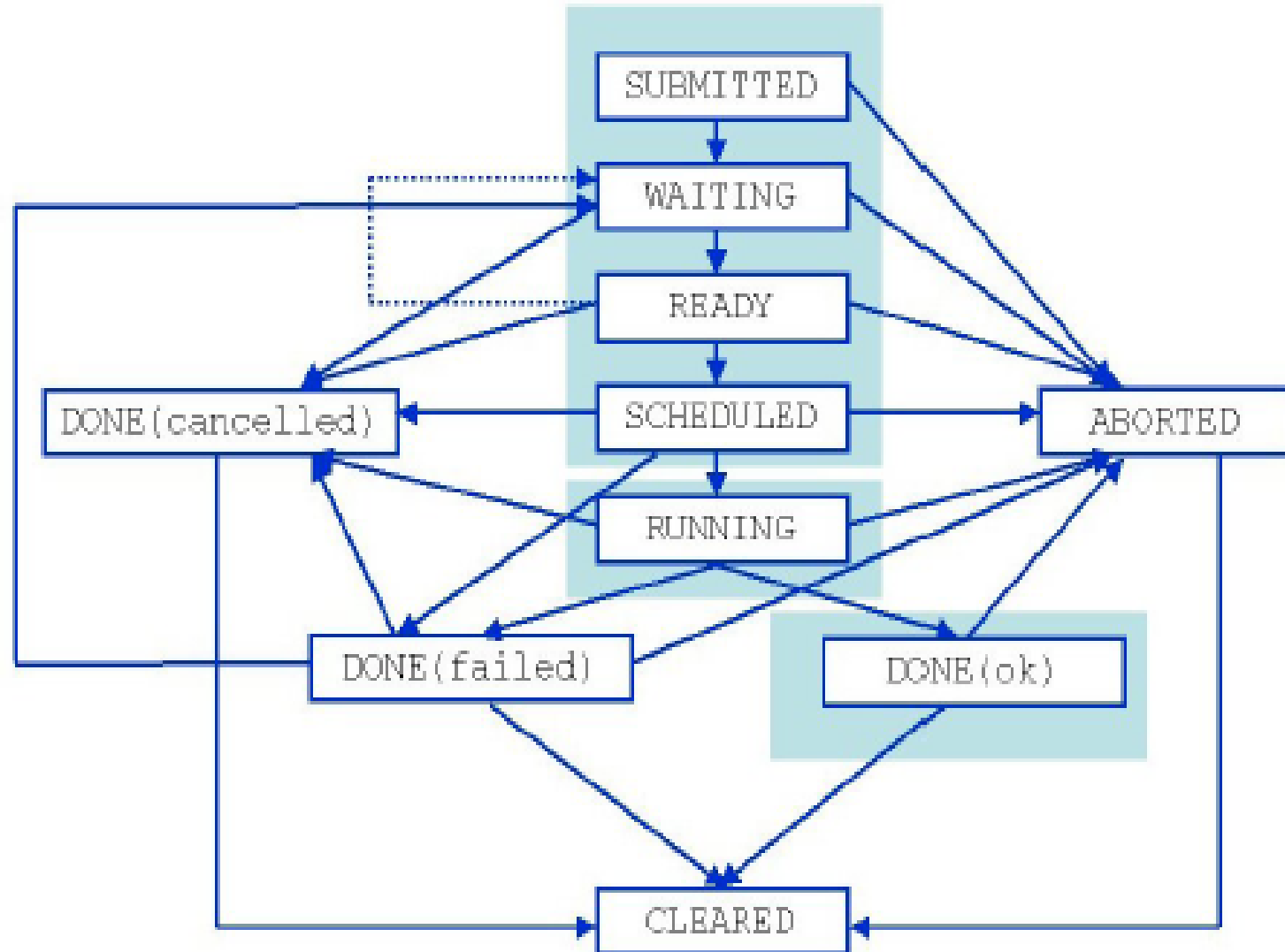
# WMS Job Submission Services



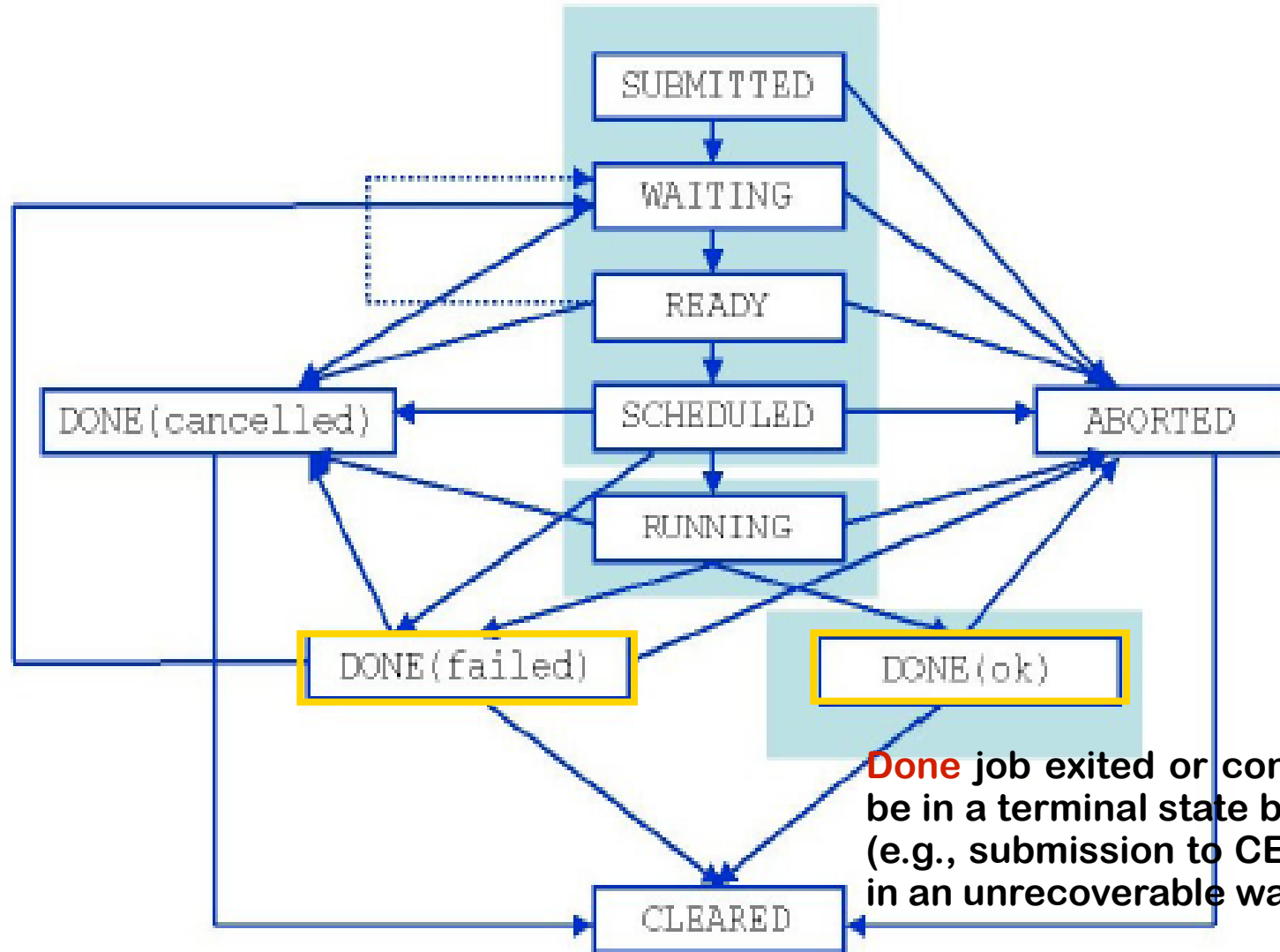
**Running** job is running.



# WMS Job Submission Services

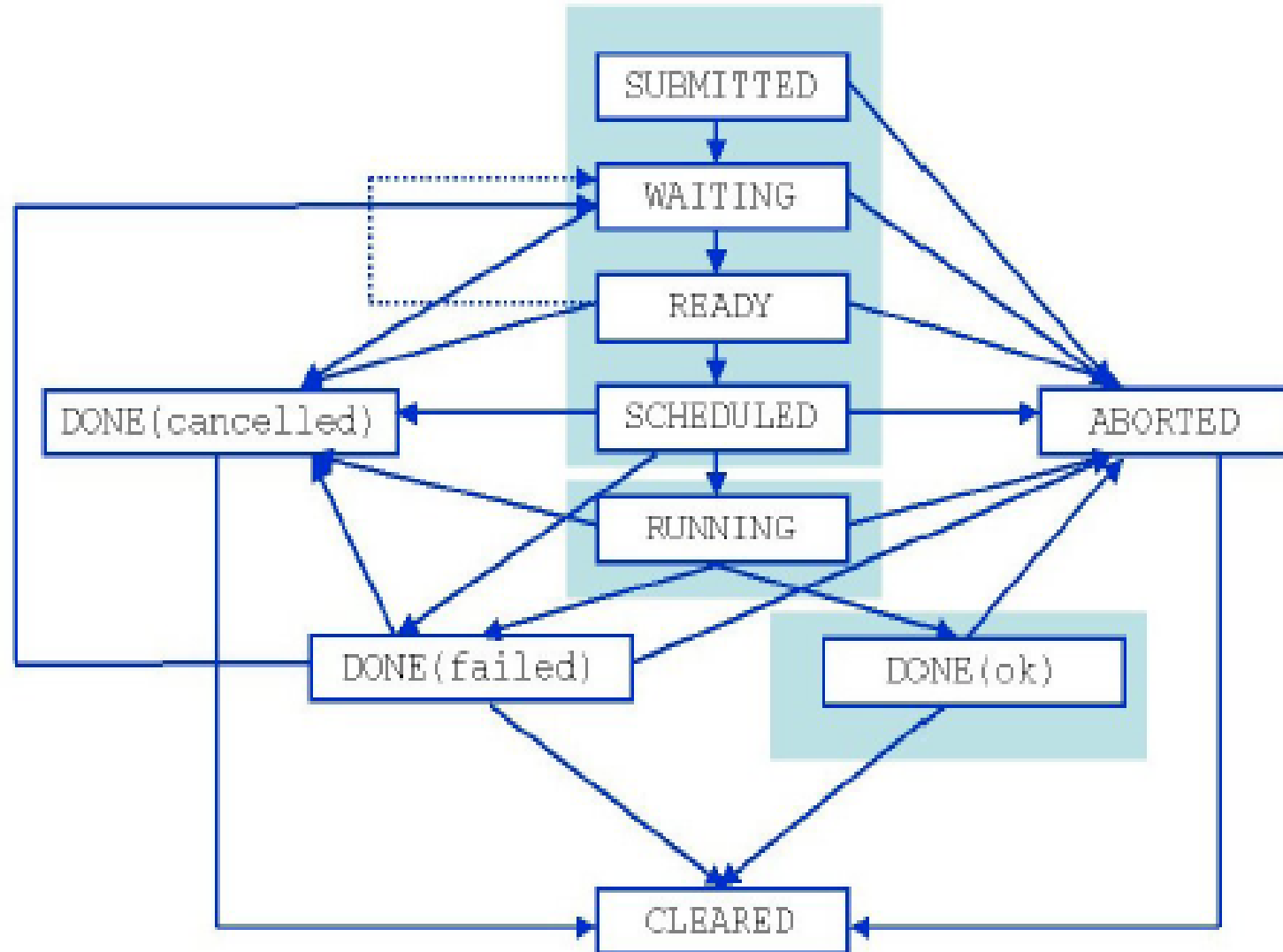


# WMS Job Submission Services



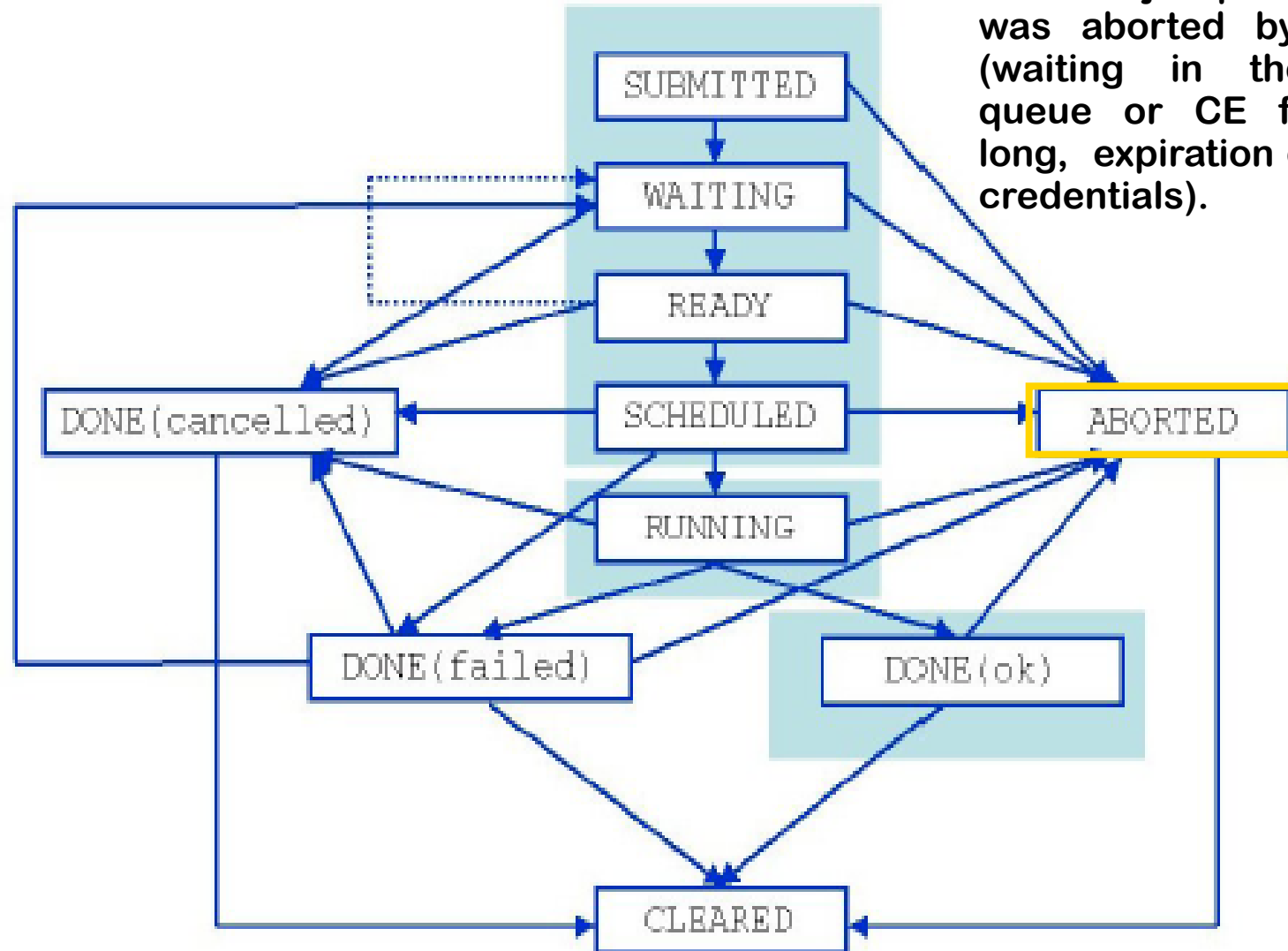
**Done** job exited or considered to be in a terminal state by CondorC (e.g., submission to CE has failed in an unrecoverable way).

# WMS Job Submission Services

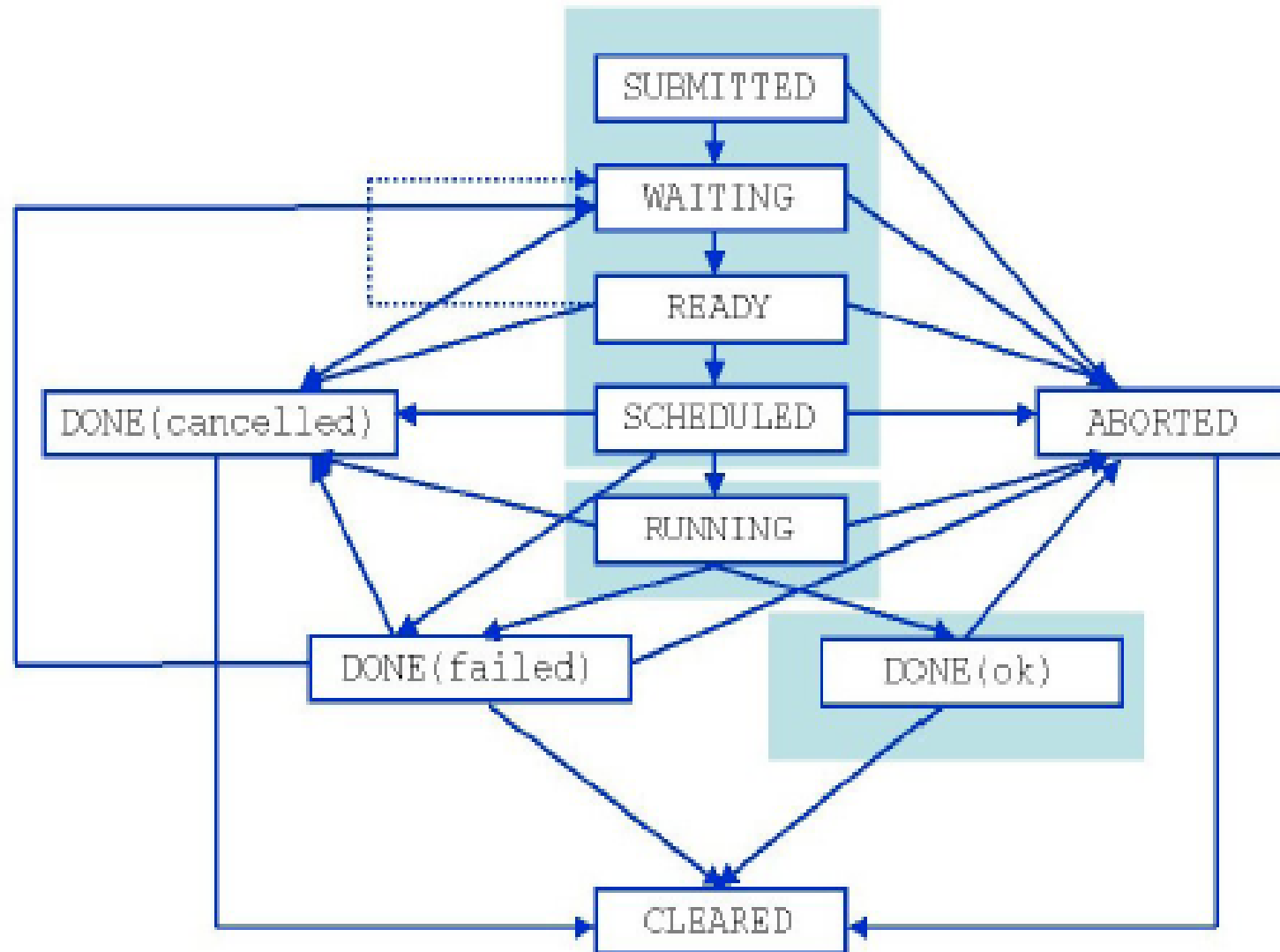


# WMS Job Submission Services

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

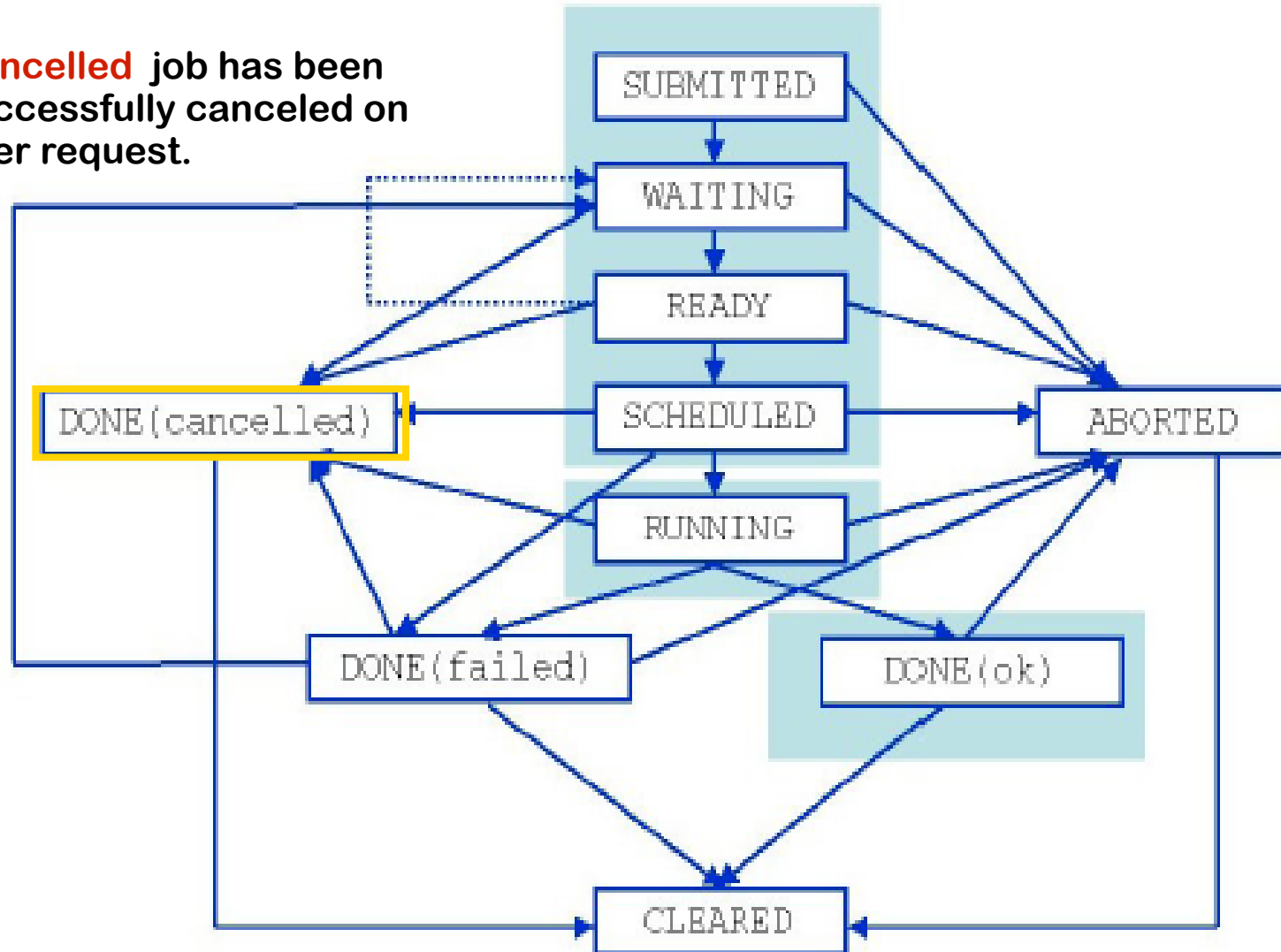


# WMS Job Submission Services

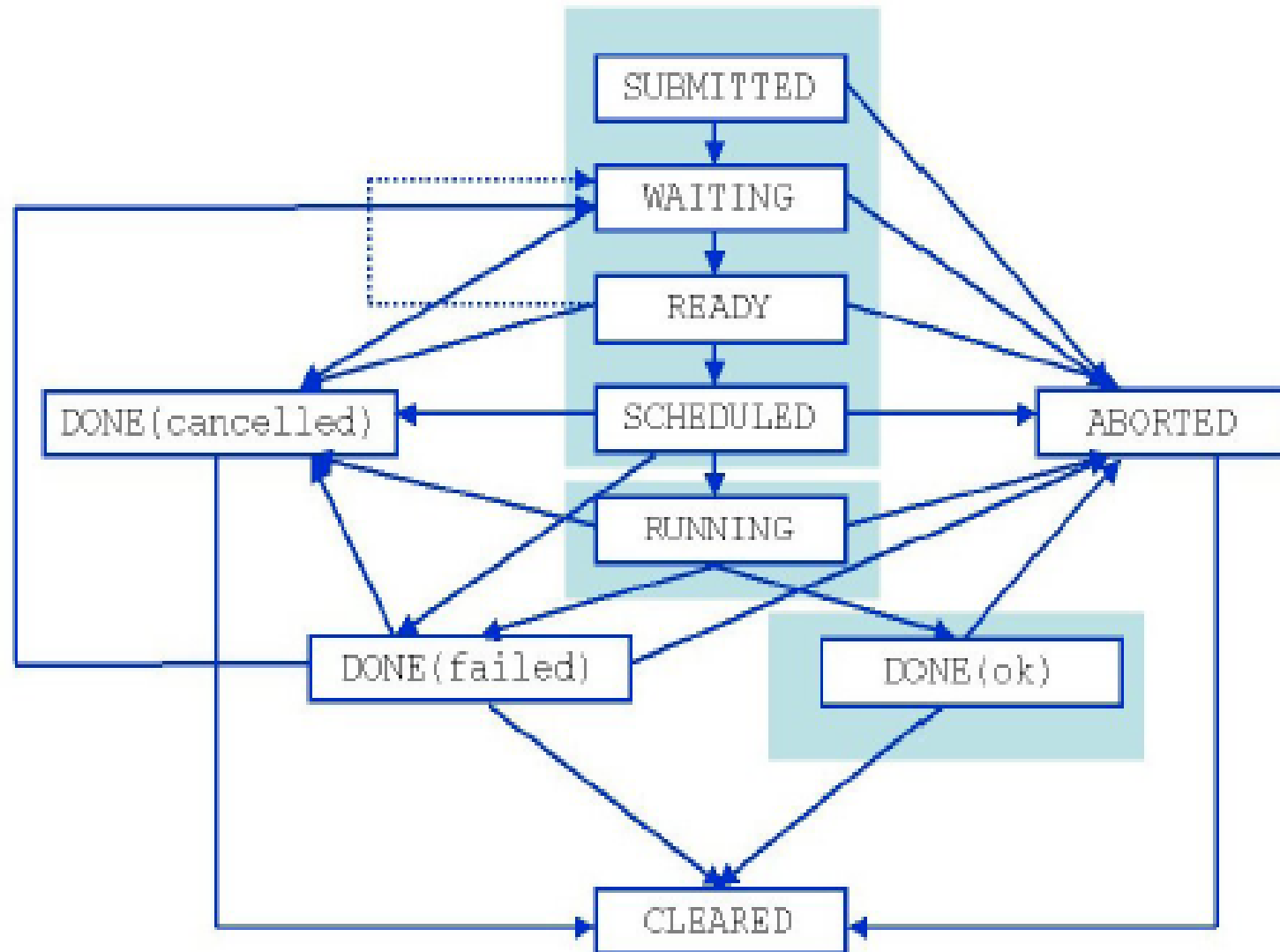


# WMS Job Submission Services

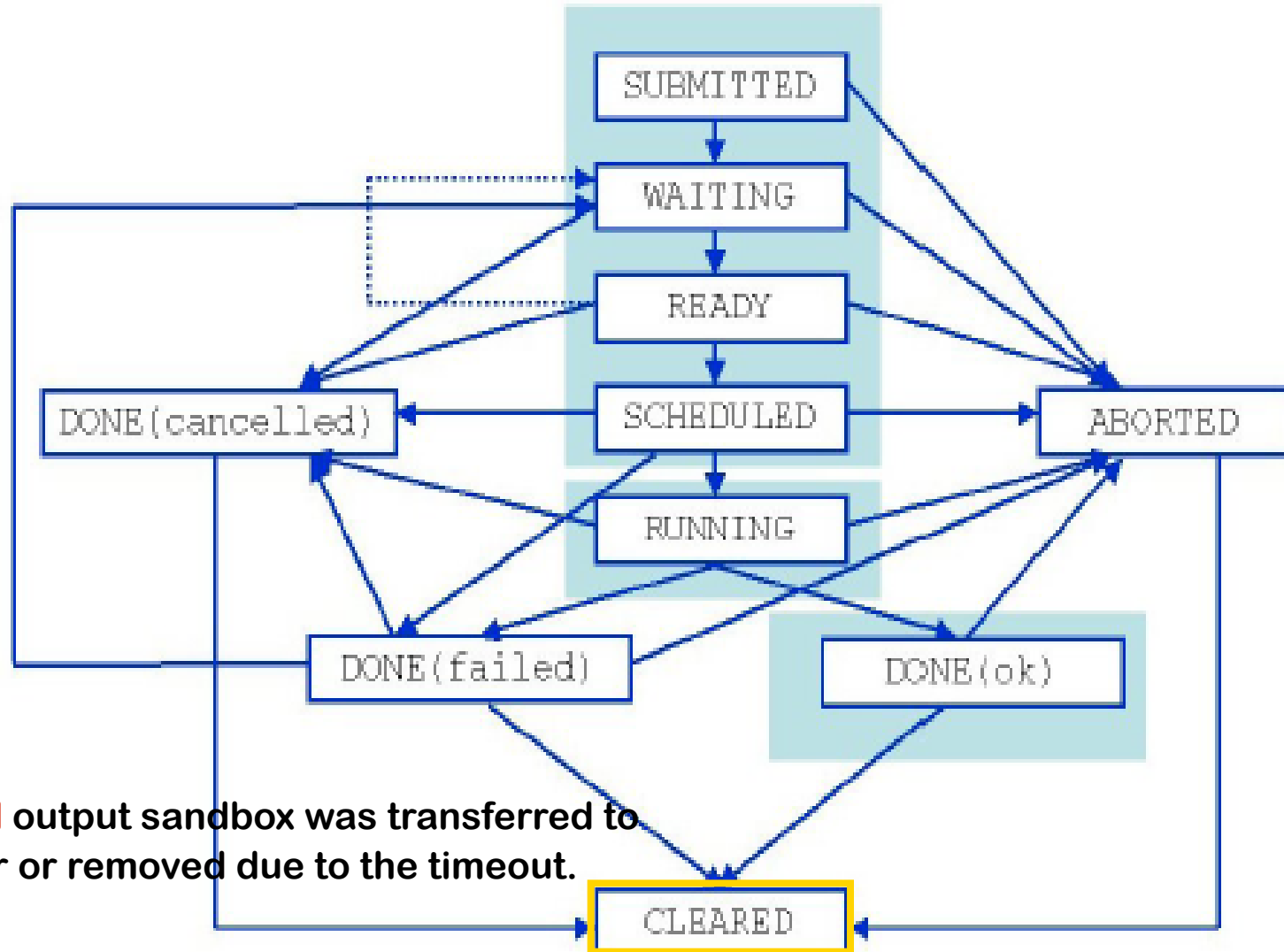
**Cancelled** job has been successfully canceled on user request.



# WMS Job Submission Services



# WMS Job Submission Services



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



# Service architecture

# Service architecture



# Service architecture

Find the list of resources  
suitable to run a specific job

**“User  
interface”**



**“possible  
operations”**

# Service architecture

Find the list of resources  
suitable to run a specific job

Submit a job/DAG for  
execution on a remote  
Computing Element

**“User  
interface”**



**“possible  
operations”**

# Service architecture

Find the list of resources  
suitable to run a specific job

Submit a job/DAG for  
execution on a remote  
Computing Element

Check the status of a  
submitted job/DAG

**“User  
interface”**



**“possible  
operations”**

# Service architecture

Find the list of resources  
suitable to run a specific job

Submit a job/DAG for  
execution on a remote  
Computing Element

Check the status of a  
submitted job/DAG

Cancel one or more  
submitted jobs/DAGs

**“User  
interface”**



**“possible  
operations”**

# Service architecture

Find the list of resources  
suitable to run a specific job

**“User  
interface”**



**“possible  
operations”**

Retrieve checkpoint states of  
a submitted checkpointable job

Submit a job/DAG for  
execution on a remote  
Computing Element

Check the status of a  
submitted job/DAG

Cancel one or more  
submitted jobs/DAGs

# Service architecture

**“User interface”**



**“possible operations”**

Find the list of resources suitable to run a specific job

Submit a job/DAG for execution on a remote Computing Element

Retrieve checkpoint states of a submitted checkpointable job

Retrieve the output files of a completed job/DAG (output sandbox)

Check the status of a submitted job/DAG

Cancel one or more submitted jobs/DAGs



# Service architecture

**“User interface”**



**“possible operations”**

Find the list of resources suitable to run a specific job

Submit a job/DAG for execution on a remote Computing Element

Check the status of a submitted job/DAG

Cancel one or more submitted jobs/DAGs

Retrieve checkpoint states of a submitted checkpointable job

Retrieve the output files of a completed job/DAG (output sandbox)

Retrieve and display bookkeeping information about submitted jobs/DAGs

# Service architecture

**“User interface”**



**“possible operations”**

Find the list of resources suitable to run a specific job

Submit a job/DAG for execution on a remote Computing Element

Check the status of a submitted job/DAG

Cancel one or more submitted jobs/DAGs

Retrieve checkpoint states of a submitted checkpointable job

Retrieve the output files of a completed job/DAG (output sandbox)

Retrieve and display bookkeeping information about submitted jobs/DAGs

Retrieve and display logging information about submitted jobs/DAGs

# Service architecture

**“User interface”**



**“possible operations”**

Find the list of resources suitable to run a specific job

Submit a job/DAG for execution on a remote Computing Element

Check the status of a submitted job/DAG

Cancel one or more submitted jobs/DAGs

Retrieve checkpoint states of a submitted checkpointable job

Retrieve the output files of a completed job/DAG (output sandbox)

Retrieve and display bookkeeping information about submitted jobs/DAGs

Retrieve and display logging information about submitted jobs/DAGs

Start a local listener for an interactive job

# Command Line Interface

- The most relevant commands to interact with the WMS (NS):
  - `edg-job-submit <jdl_file>`
  - `edg-job-list-match <jdl_file>`
  - `edg-job-status <job_id>`
  - `edg-job-get-output <job_id>`
  - `edg-job-cancel <job_id>`
- In gLite 3.0:



# Command Line Interface

- **Job Submission**

- Perform the job submission to the Grid.

`$ edg-job-submit [options] <jdl_file>`

`$ glite-job-submit [options] <jdl_file>`



- where <jdl file> is a file containing the job description, usually with extension .jdl.

# Command Line Interface

If the request has been correctly submitted this is the typical output that you can get:

**edg-job-submit test.jdl**

=====glite-job-submit Success =====

The job has been successfully submitted to the Network Server.

Use edg-job-status command to check job current status.

Your job identifier (edg\_jobId) is:

- <https://lxshare0234.cern.ch:9000/rIBubkFFKhnsQ6CjiLUY8Q>

=====

In case of failure, an error message will be displayed instead, and an exit status different from zero will be returned.

# Command Line Interface

It is possible to see which CEs are eligible to run a job specified by a given JDL file using the command

**edg-job-list-match test.jdl**

**Connecting to host lxshare0380.cern.ch, port 7772**

**Selected Virtual Organisation name (from UI conf file): dteam**

\*\*\*\*\*

## **COMPUTING ELEMENT IDs LIST**

**The following CE(s) matching your job requirements have been found:**

**adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite**

**adc0015.cern.ch:2119/jobmanager-lcgpbs-long**

**adc0015.cern.ch:2119/jobmanager-lcgpbs-short**

\*\*\*\*\*

# Command Line Interface

After a job is submitted, it is possible to see its status using the `glite-job-status` command.

`edg-job-status` <https://lxshare0234.cern.ch:9000/X-ehTxfdlXxSoIdVLS0L0w>

\*\*\*\*\*

## BOOKKEEPING INFORMATION:

Printing status info for the Job:

`https://lxshare0234.cern.ch:9000/X-ehTxfdlXxSoIdVLS0L0w`

Current Status: Scheduled

Status Reason: unavailable

Destination: `lxshare0277.cern.ch:2119/jobmanager-pbs-infinite`

reached on: Fri Aug 1 12:21:35 2003

\*\*\*\*\*



# Command Line Interface

A job can be canceled before it ends using the command `glite-job-cancel`.

`edg-job-cancel` <https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog>

Are you sure you want to remove specified job(s)? [y/n]n :y

===== glite-job-cancel Success=====

The cancellation request has been successfully submitted for the following job(s)

- <https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog>

=====

# Command Line Interface

After the job has finished (it reaches the DONE status), its output can be copied to the UI

**edg-job-get-output** <https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg>

Retrieving files from host **lxshare0234.cern.ch**

\*\*\*\*\*

**JOB GET OUTPUT OUTCOME**

Output sandbox files for the job:

- <https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg>

have been successfully retrieved and stored in the directory:

**/tmp/jobOutput/snPegp1YMJcnS22yF5pFlg**

\*\*\*\*\*

By default, the output is stored under **/tmp**, but it is possible to specify in which directory to save the output using the **- -dir <path name>** option.

# Second Part

## Job Description Language

# Job Description Language

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.

# Job Description Language

- 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.

# Job Description Language

- ✚ In a JDL, some attributes are mandatory while others are optional.
- ✚ An “essential” JDL is the following:

```
Executable = “test.sh”;  
StdOutput = “std.out”;  
StdError = “std.err”;  
InputSandbox = {“test.sh”};  
OutputSandbox = {“std.out”, “std.err”};
```

- ✚ If needed, arguments to the executable can be passed:

```
Arguments = “Hello World!”;
```

- ✚ If the argument contains quoted strings, the quotes must be escaped with a backslash

e.g. Arguments = “\”Hello World!\“ 10”;

- ✚ Special characters such as **&**, **|**, **>**, **<** are only allowed if specified inside a quoted

# Workload Manager Service

- The JDL allows the description of the following request types supported by the WMS:

**Job: a simple application**

**DAG: a direct acyclic graph of dependent jobs**

- With WMSPoxy

**Collection: a set of independent jobs**

- With WMSPoxy



# Jobs

## •The Workload Management System currently supports the following types for Jobs :

- **Normal** a simple batch, a set of commands to be processed as single unit
- **Interactive** a job whose standard streams are forwarded to the submitting client
- **MPICH** a parallel application using MPICH-P4 implementation of MPI
- **Partitionable** a job which is composed by a set of independent steps/ iterations
- **Checkpointable** a job able to save its state
- **Parametric** a job where one or more of its attributes are parameterized

Support for MPI and parametric jobs is only available when the submission to the WMS is done through the WMPProxy service

# Jobs

## •The Workload Management System currently supports the

a set of independent sub-jobs, each one taking care of a step or of a sub-set of steps, and which can be executed in parallel

Jobs :

• **Batch** a set of commands to be processed as single

• **Standard** those standard streams are forwarded to the submitting client

- **MPICH** a parallel application using MPICH-P4 implementation of MPI
- **Partitionable** a job which is composed by a set of independent steps/iterations
- **Checkpointable** a job able to save its state
- **Parametric** a job where one or more of its attributes are parameterized

Support for MPI and parametric jobs is only available when the submission to the WMS is done through the WMPProxy service

# Jobs

## •The Workload Management System currently supports the

a set of independent sub-jobs, each one taking care of a step or of a sub-set of steps, and which can be executed in parallel

jobs :

batch, a set of commands to be processed as single

those standard streams are for

submitting client

- **MPICH** a parallel application using MPICH-P4 in
- **Partitionable** a job which is composed by a set of iterations
- **Checkpointable** a job able to save its state
- **Parametric** a job where one or more of its attributes are parameterized

the job execution can be suspended and resumed later, starting from the same point where it was first stopped

Support for MPI and parametric jobs is only available when the submission to the WMS is done through the WMPProxy service

# JDL: Relevant Attributes

## + JobType (optional)

● Normal (simple, sequential job), Interactive,  
MPICH, Checkpointable, Partitionable,  
Parametric





● Or combination of them

● Checkpointable, Interactive

● Checkpointable, MPI

E.g. JobType = "Interactive";  
JobType =  
{ "Interactive", "Checkpointable" };

## **Executable (mandatory)**

-  This is a string representing the executable/ command name.
-  The user can specify an executable which is already on the remote CE
-  Executable = {"/opt/EGEODE/GCT/egeode.sh"};
-  The user can provide a local executable name, which will be staged from the UI to the WN.  
Executable = {"egeode.sh"};  
InputSandbox = {"/home/larocca/egeode/egeode.sh"};

## **Arguments (optional)**

This is a string containing all the job command line arguments.

**E.g.:** If your executable sum has to be started as:

```
$ sum N1 N2 -out result.out
```

## + Environment (optional)

- List of environment settings needed by the job to run properly

E.g. Environment = {"JAVA\_HOME=/usr/java/j2sdk1.4.2\_08"};

## + InputSandbox (optional)

- List of files on the UI local disk needed by the job for proper running
- The listed files will be automatically staged to the remote resource

E.g. InputSandbox = {"myscript.sh", "/tmp/cc.sh"};




## **OutputSandbox (optional)**

-  List of files, generated by the job, which have to be retrieved from the CE

E.g. `OutputSandbox = { "std.out", "std.err",  
"image.png" };`



## **Requirements (optional)**

-  Job requirements on computing resources
-  Specified using attributes of resources published in the Information Service
-  If not specified, default value defined in UI configuration file is considered

**Default. Requirements =**  
**other.GlueCEStateStatus == "Production";**

**Requirements=other.GlueCEUniqueID == "adc006.cern.ch:2119/  
 jobmanager-pbs-infinite"**

**Requirements=Member("ALICE-3.07.01",  
 other.GlueHostApplicationSoftwareRunTimeEnvironment);**

# References

## JDL Attributes

[http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0142-0\\_2.pdf](http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0142-0_2.pdf)

<https://edms.cern.ch/document/590869/1>

[http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/api\\_doc/wms\\_jdl/index.html](http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/api_doc/wms_jdl/index.html)

- LCG-2 User Guide Manual Series

<https://edms.cern.ch/file/454439/LCG-2-UserGuide.html>

# Third Part

## Workload Manager Proxy

- **WMPProxy (Workload Manager Proxy)**
  - is a new service providing access to the gLite Workload Management System (WMS) functionality through a simple Web Services based interface.
  - has been designed to handle a large number of requests for job submission
    - gLite 1.5 => ~180 secs for 500 jobs
    - goal is to get in the short term to ~60 secs for 1000 jobs
  - it provides additional features such as *bulk submission* and the support for *shared and compressed* sandboxes for *compound jobs*.
  - It's the natural replacement of the NS in the passage to the *SOA approach*.

- **Support for new types strongly relies on newly developed JDL converters and on the DAG submission support**
  - all JDL conversions are performed on the server
  - a single submission for several jobs
- **All new request types can be monitored and controlled through a single handle (the request id)**
  - each sub-jobs can be however followed-up and controlled independently through its own id
- **“Smarter” WMS client commands/API**
  - allow submission of DAGs, collections and parametric jobs exploiting the concept of “shared sandbox”
  - allow automatic generation and submission of collections and DAGs from sets of JDL files located in user specified directories on the UI

# WMPProxy C++ client commands

The commands to interact with WMPProxy Service are:

**glite-wms-job-submit** <jdl\_file>

**glite-wms-job-list-match** <jdl\_file>

**glite-wms-job-cancel** <job\_ids>

In our examples:  
**glite-wms-job-\***  
are  
**edg-job-\***

- **gLite 3.0 User Guide**
  - <https://edms.cern.ch/file/722398/1.1/gLite-3-UserGuide.pdf>
- **R-GMA overview page**
  - <http://www.r-gma.org/>
- **GLUE Schema**
  - <http://infnforge.cnaf.infn.it/glueinfomodel/>
- **JDL attributes specification for WM proxy**
  - <https://edms.cern.ch/document/590869/1>
- **WMProxy quickstart**
  - [http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wmproxy\\_client\\_quickstart.shtml](http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wmproxy_client_quickstart.shtml)
- **WMS user guides**
  - <https://edms.cern.ch/document/572489/1>

# Questions...

