

Glite WMS overview

Alessandra Forti

Computing Seminar

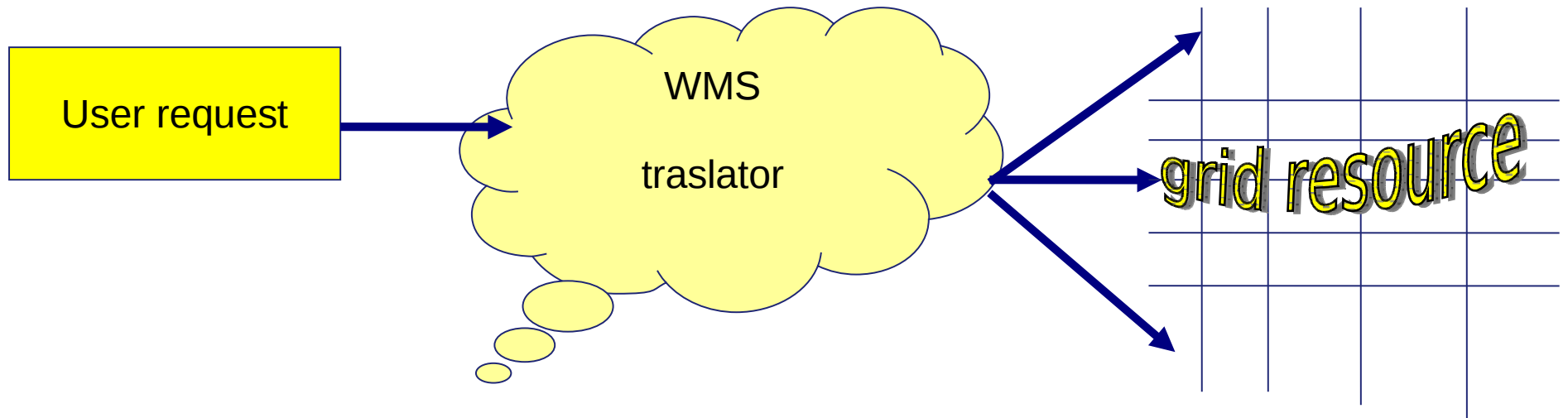
Manchester

20th November 2008

Talk Overview

- Architecture
- A bit of history
- WMS definition
- Components description
- What you can do from a UI
- Command line names
- Simple examples
- Job lifetime
- Forget the RB
- WMPProxy
- Delegation
- Proxy Renewal
- New Features
- Documentation
- Questions

Architecture: Simple view



A bit of History

There is only one software product, which has been in continuous development since before the start of EDG, so about 8 years now. The developers always called it the WMS, the broker is just one component within it, but the RB name managed to stick in common use. When LCG was set up in 2003 they took the then-current release version for the LCG 1 production service, but they found that the performance was a long way short of production quality, so they put in a lot of effort to make it more robust. However, they didn't evolve the functionality, and it has had minimal maintenance for several years now. Meanwhile the developers kept developing, as developers tend to do. They produced a major new version in 2004, which was incorporated into the glite 1.x release series, which were however never deployed in production. In 2006 the glite and LCG releases were forcibly merged as glite 3.0 and the WMS did go into production, largely for political reasons, but the performance was poor and it was largely unused. The developers carried on developing, and produced another major upgrade intended for the glite 3.1 release series. However, with the advent of SA3 in EGEE II the acceptance criteria for certification became realistic - which is to say, much, much tougher. Most of the certification was of course done at Imperial. Now it has finally been certified as production-quality, hence the old RB can finally be retired, a mere three years later than originally intended. Of course. Development still goes on, the next significant thing will be support for submission to CREAM. One hopes that certification can go a bit faster from now on ...

(S. Burke)

WMS Definition

- Work Load Management System (WMS) is a set of middleware components whose task is to distribute and manage jobs across grid resources.
- In particular the role of the Workload Manager (WM) is to satisfy requests of job management coming from its clients.
 - i.e. The WM will pass a job to a Computing Element (CE) satisfying the requirements in the job description.
- The selection of the CE is the result of the MatchMaking process.

Components

- The **Network Server (NS)** is a generic network daemon that provides job control functionality. It is responsible for accepting requests from the UI and passing them to the Workload Manager. It still works for the RB but has been obsoleted in the WMS framework.
- The **Workload Manager Proxy (WMPProxy)** provides the same job control functionality as the NS but through a Web Services based interface on top of which there are new functionalities that I will explain a bit later.

Components

- **Workload Manager (WM)** is the core component that talks to all the others to manage the jobs. It's a sort of message passing mechanism
- **Resource Broker (RB)** is responsible for the match making process. In this context is just a software component not a server.
- **Information Super Market (ISM)** is a cache of all the information necessary to do the matchmaking. Is dynamically updated by the **Information Updater** through a mixture of polling resources and receiving notifications.

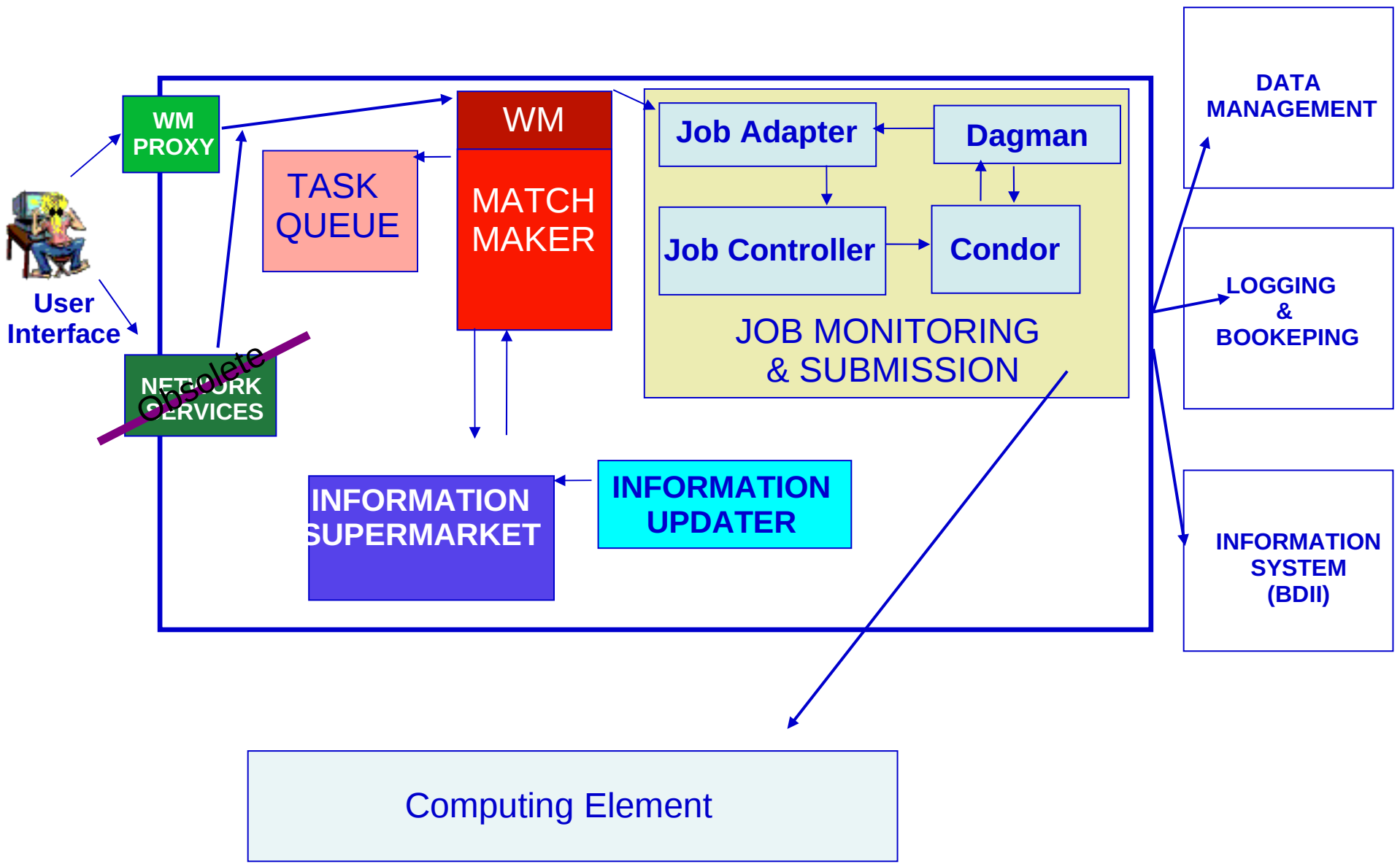
Components

- **Task Queue** allows the jobs to be queued in the system in case there are no resources available. The scheduling of the jobs can then be
 - **Active:** i.e. The job is resubmitted every so often until a resource becomes available and the submission succeeds
 - **Passive:** i.e. The scheduler waits for a resource that matches the job requirements before resubmitting it.

Components

- **MyProxy** is a service to renew the user proxy to guarantee that the life time of the proxy covers the life time of the jobs and the jobs don't fail due to proxy expiration.
- **Logging&Bookkeeping (LB)** is a service that logs all the events of all the jobs managed by a WM. It is the service contacted by the users when they want to know the status of their jobs.

Architecture: complicated view



What you can do from the UI

Find the list of resources that satisfy your jobs requirements

Retrieve the output of the completed jobs

Retrieve checkpoint information of checkpointable jobs

Submit single or compound jobs to the system

Retrieve and display bookkeeping information of submitted jobs

Start a local listener for an interactive job

Check the jobs status

Retrieve and display logging information of submitted jobs

Cancel jobs

Command line names: RB/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>
- edg-job-get-logging-info <job_id>

OBSOLETE!!!

Command line names: WMS/NS

- `glite-job-submit <jdl_file>`
- `glite-job-list-match <jdl_file>`
- `glite-job-status <job_id>`
- `glite-job-output <job_id>`
- `glite-job-cancel <job_id>`
- `glite-job-get-logging-info <job_id>`

OBSOLETE!!! (never used really)

Command line names: WMS/WMPProxy

- `glite-wms-delegate-proxy -d <deleg_id>`
- `glite-wms-job-submit -d <deleg_id> <jdl_file>`
- `glite-wms-job-list-match <jdl_file>`
- `glite-wms-job-status <job_id>`
- `glite-wms-job-output <job_id>`
- `glite-wms-job-cancel <job_id>`
- `glite-wms-job-get-logging-info <job_id>`

CURRENT

Simple Example

- Create your proxy
 - `voms-proxy-init -voms atlas`
- Create the delegation
 - `glite-wms-delegate-proxy -d pippo3`
- Submit the job with the delegated credentials
 - `glite-job-submit -d pippo3 my_test.jdl`
- Check the status of your job
 - `glite-job-status`
https://lcgwms02.gridpp.ac.uk:9000/LHGIagvDl701_msz0jpIg
- Get the output
 - `glite-job-output`
https://lcgwms02.gridpp.ac.uk:9000/LHGIagvDl701_msz0jpIg

Simple Example jdl

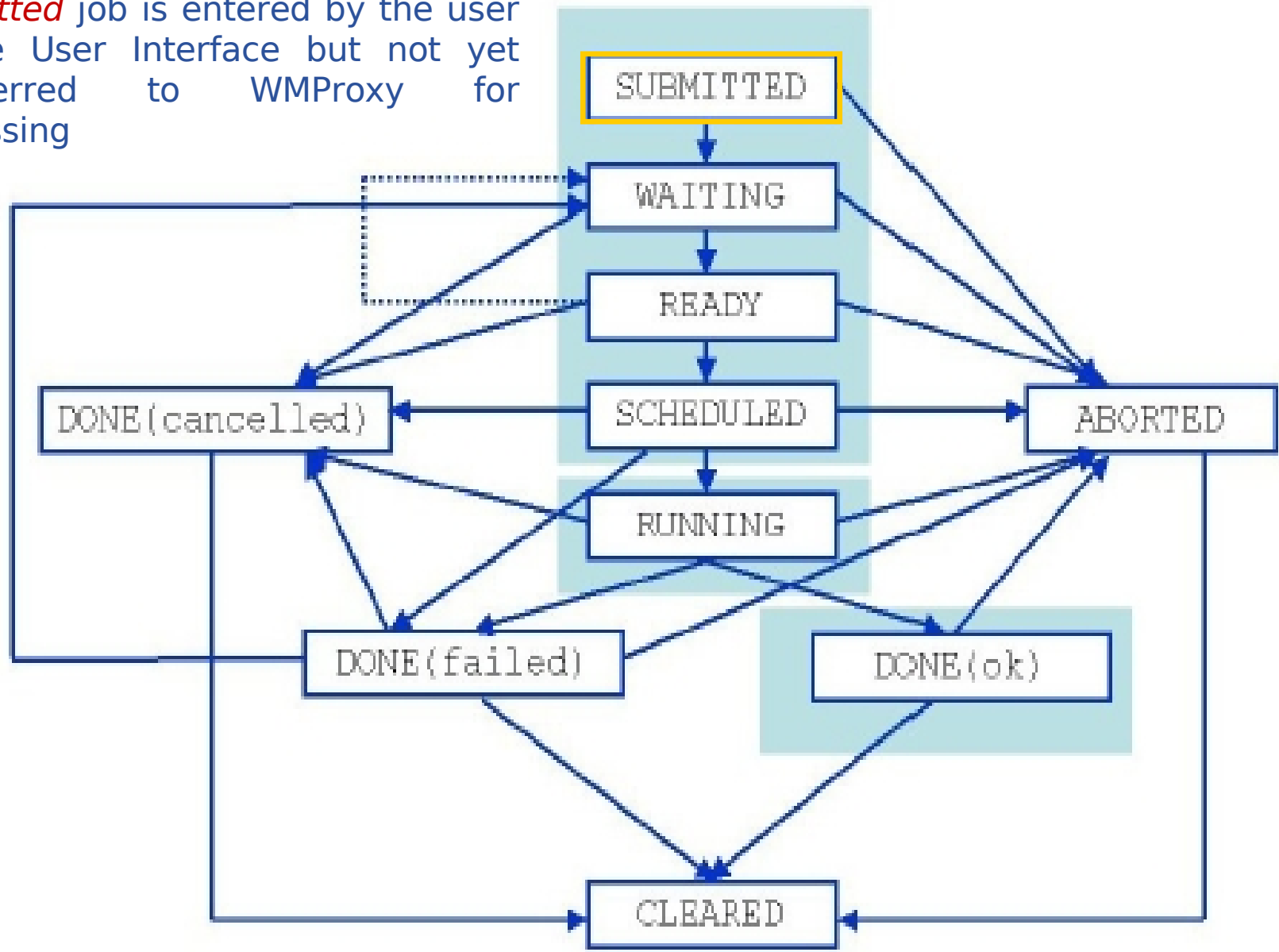
- Simple JDL file

```
Executable = "my_script.sh";  
StdOutput = "stdout.log";  
StdError = "stderr.log";  
InputSandbox = { "test.sh" }  
OutputSandbox = {"stdout.log","stderr.log"}  
Arguments = "-v 3"  
Requirements = other.GlueCEUniqueID ==  
"ce01.tier2.hep.manchester.ac.uk:2119/jobmanager-lcgpbs-atlas";
```

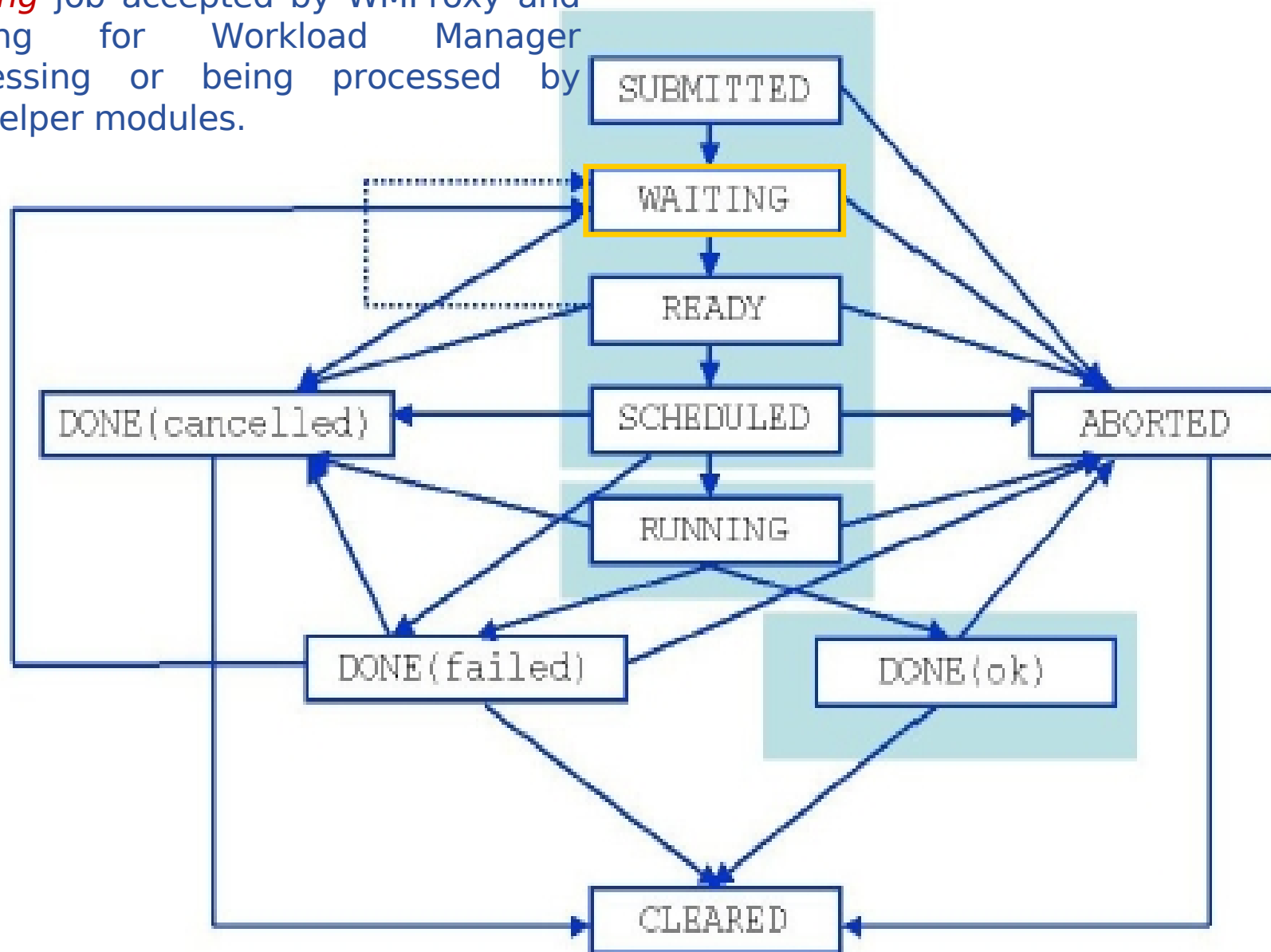
- Some default parameters set on the UI

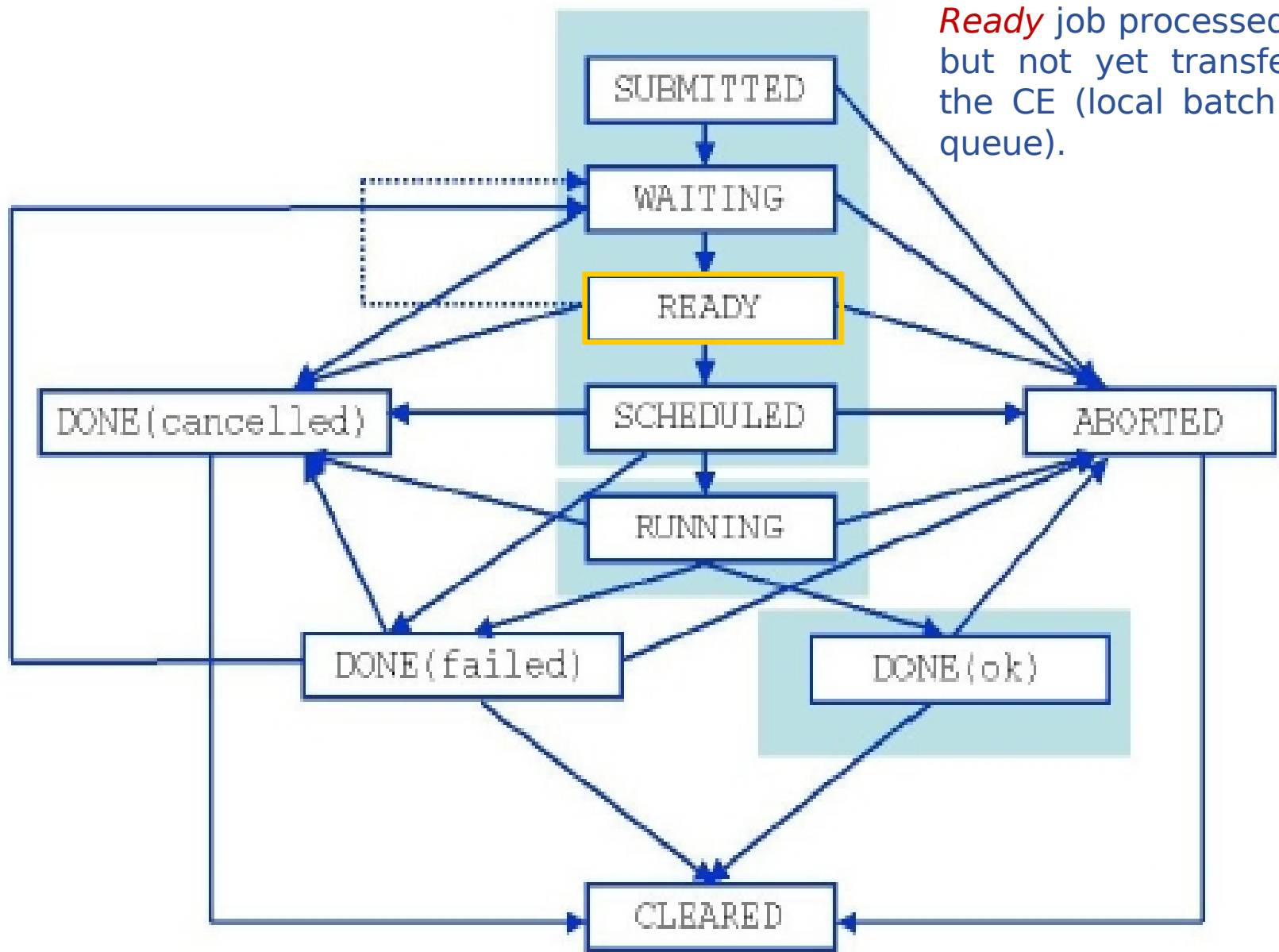
```
Requirements = other.GlueCEStateStatus == "Production";
```

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

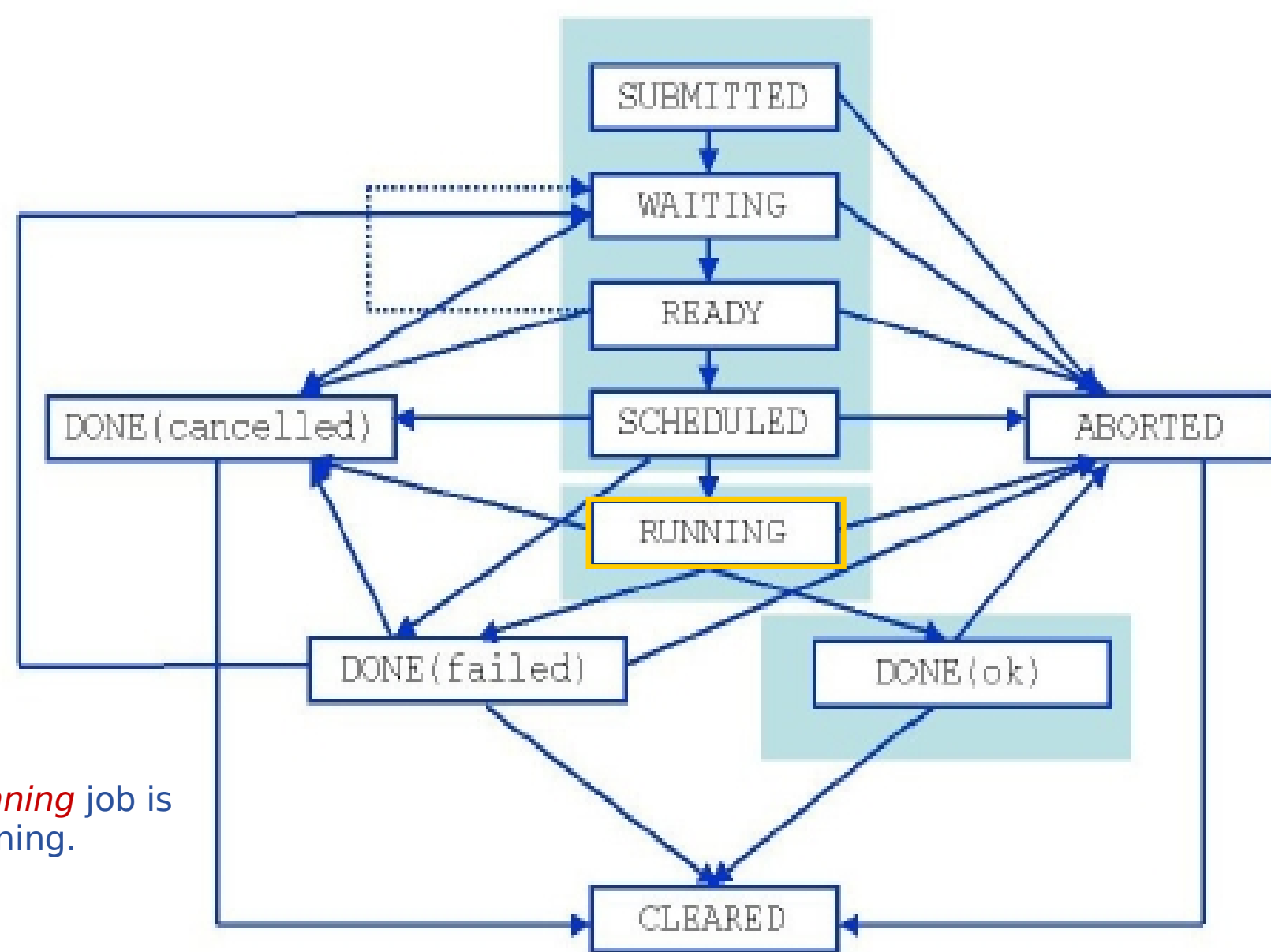


Waiting job accepted by WMPProxy and waiting for Workload Manager processing or being processed by WMHelper modules.

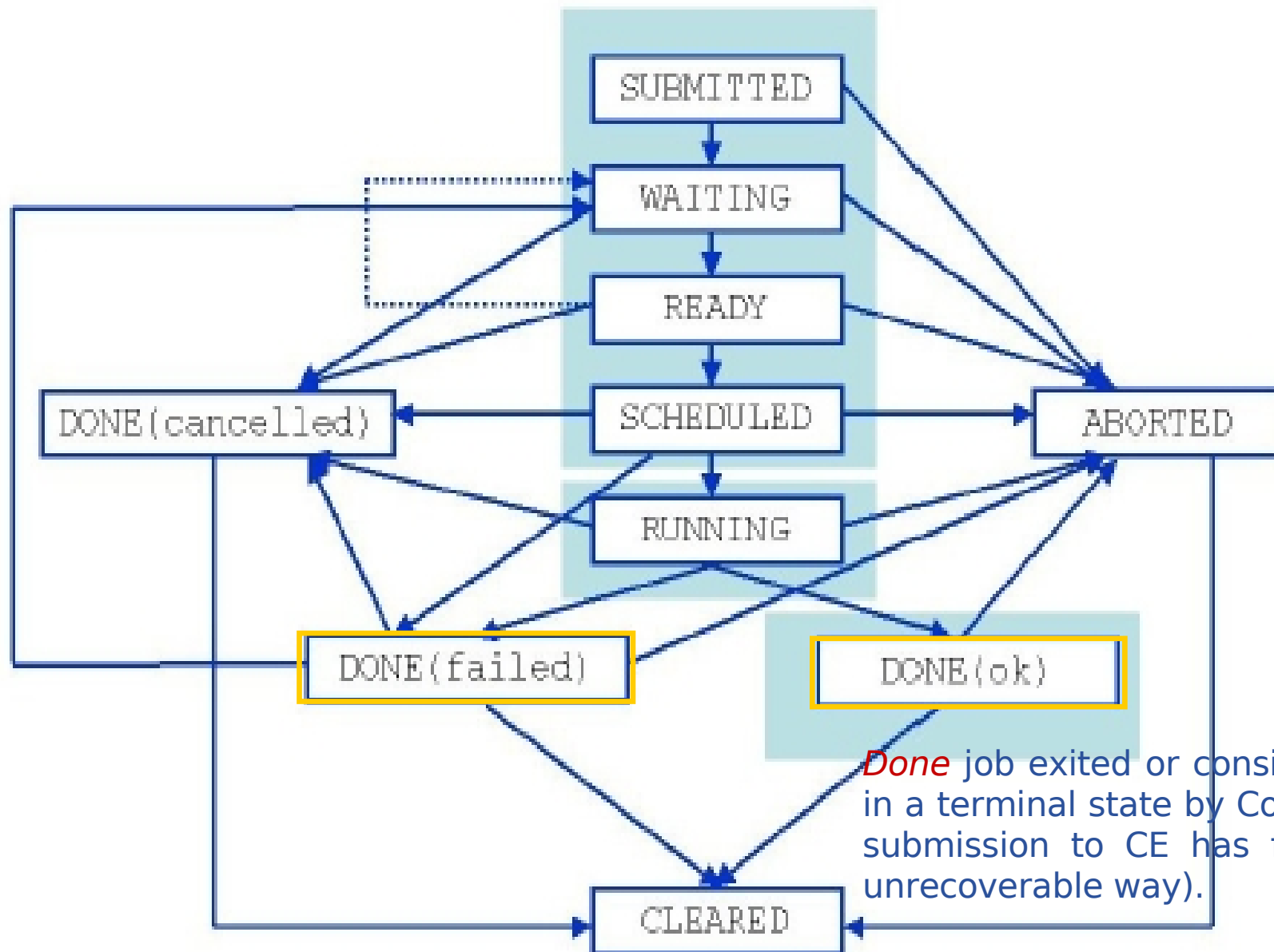




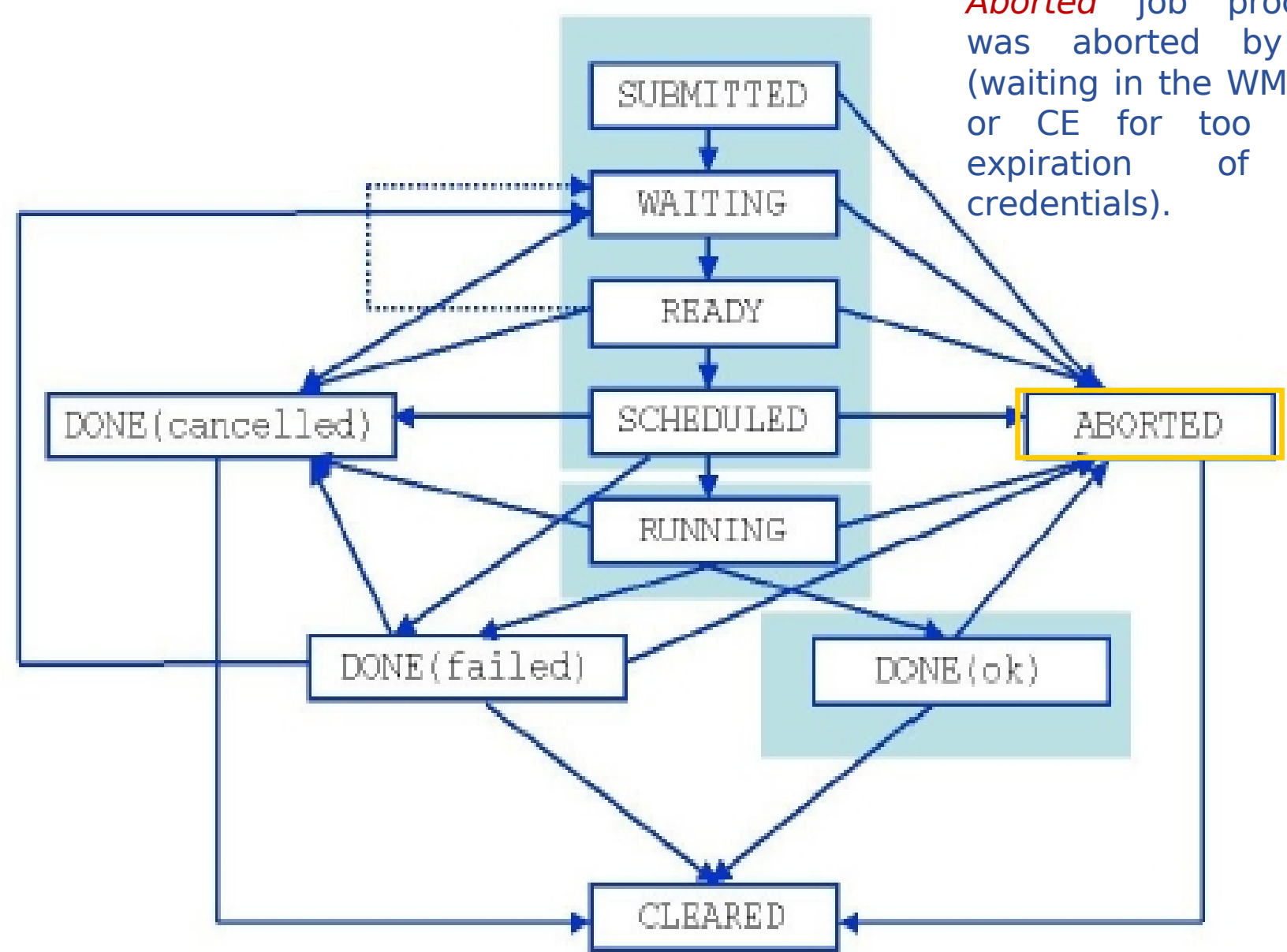
Ready job processed by WM but not yet transferred to the CE (local batch system queue).



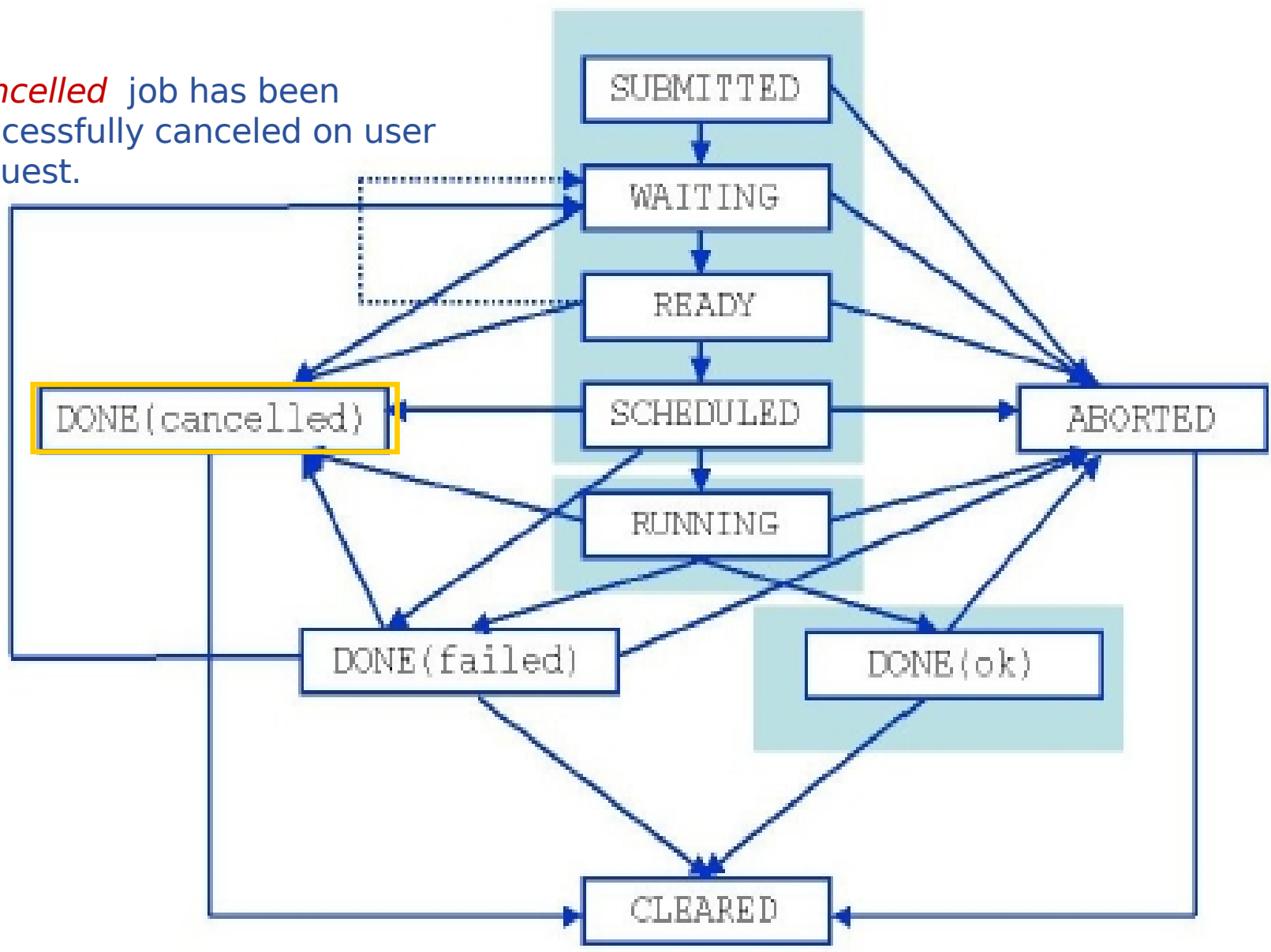
Running job is running.

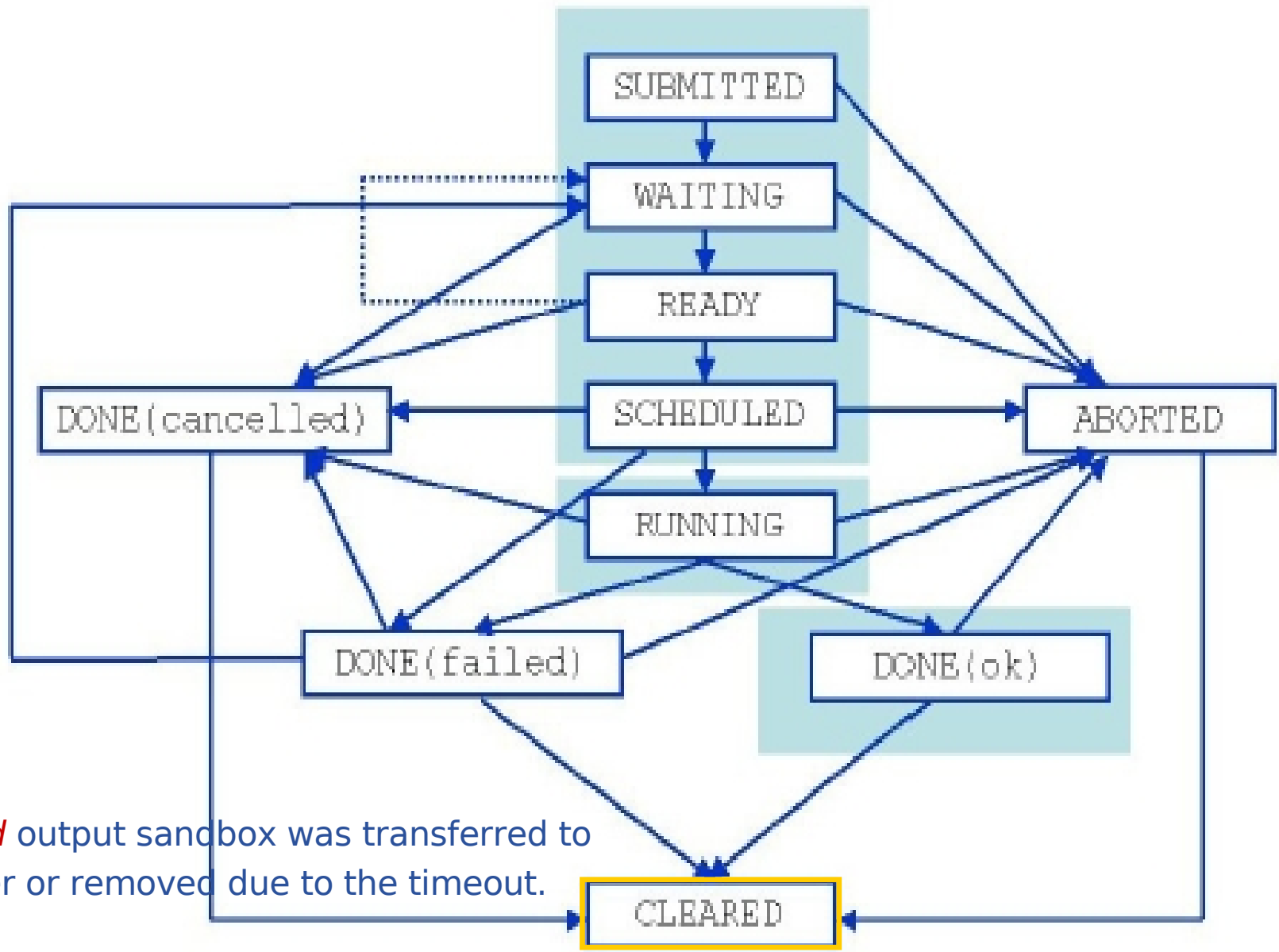


Aborted job processing was aborted by WMS (waiting in the WM queue or CE for too long, expiration of user credentials).



Cancelled job has been successfully canceled on user request.





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

Job Life Cycle

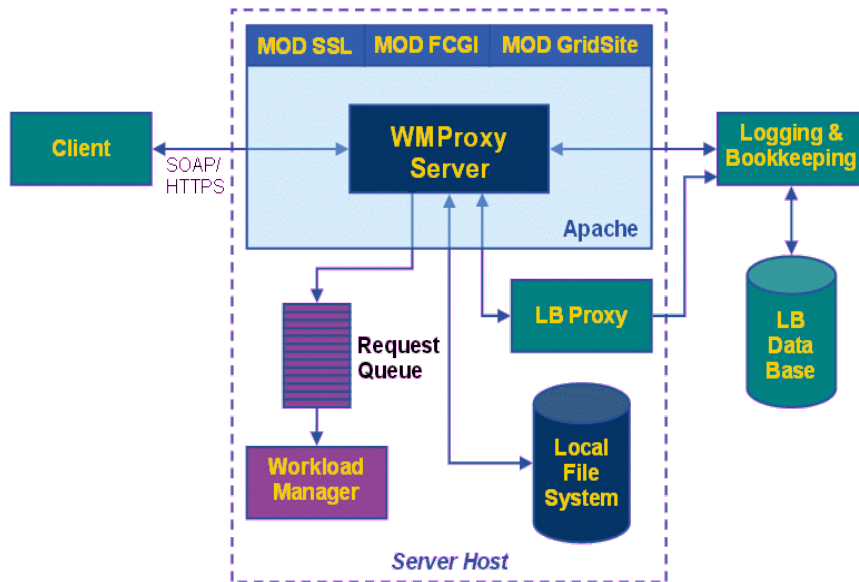
- Submitted: registered with the WMPProxy
- Waiting: waiting to be processed by WM
 - Nothing to do with the CE
- Ready: processed by WM ready to be sent to CE
- Scheduled: sent to CE
- Running: running on the CE resources
- Done:
 - Failed (many,many, many reasons)
 - Success (could be failed but the executable returns 0)
 - Canceled (by the user)
- Aborted: systems aborts the job for various reasons
- Cleared: job Done and output has been collected

Forget the RB

Feature	RB	WMS
Access Interface	network daemon NS	WEB service (WMPProxy)
Authentication	globus/lcas	gridsite/gacl
Job submission engine	globus	condor
Scalability	no numbers	10k a day for 5 days 99% eff
JDL	limited	extended
VOMS	no	yes
Delegation	no	yes
Bulk job submission	no	yes
Bulk match making	no	yes
Multiple jobs sandboxes	no	yes
Asynchronous submission	no	yes
Batch params passing	no	yes
SubCluster support	no	yes

WMPProxy

- WMPProxy is a component developed as a WEB service, to access the WMS.



- It was developed to satisfy the need to improve the performance of the existing component (NS) and to comply to the emerging standards (OGSA).
- It uses gridsite, fcgi and ssl apache modules.
 - i.e is based on a solid existing technology
 - It's in C++ rather than java.

Delegation

- Delegation means to authorize other parties to act in your place. In this case it means the user authorizes the WMS to interact with other services (like Ces, gridftp servers, catalogues) on behalf of the user.
 - A rough form of delegation, currently used, is to copy the user proxy on each machine that needs to use it to execute the job
 - WMProxy uses a delegated proxy (it signs the proxy request associated with the proxy) and stores it with an associated ID for future usage.
 - One can also delegate at each submission (automatic delegation) but it's not convenient as the signing process is CPU consuming.

Proxy Renewal

- The default lifetime of a proxy is 12 or 24 hours. Jobs often last longer and user want longer proxies. Once they could just extend the lifetime themselves through the command line, but with the advent of VOMS proxies the proxies lifetimes are set at server level.
 - Even if a user generates a 96 hours proxy the VOMS attributes within the proxy will still expire after the the time configured on the server.
- Long lived proxies get distributed everywhere and could get stolen and used without need for a passphrase as they contain also the private key.
 - Better to send around short lived proxies and renew them through an appropriate process.

Proxy Renewal

- MyProxy server is another component to which the user can transfer delegated credential. The function of this service is to refresh/renew proxies that are about to expire.
- Create a delegated proxy for MyProxy
 - `my-proxy-init -n -d`
- Create your proxy
 - `voms-proxy-init -voms atlas`
- Delegate it to WMPProxy
 - `glite-voms-delegate-proxy -d <deleg_id>`
- Tell the WMS to use the renewal service
 - `MyProxyServer` attribute in the JDL

Job types

- Single Job
 - Normal: simple batch job
 - MPICH: parallel jobs
 - Interactive: stdout and stderr streamed back to the client
- Collections *New*
 - Group of jobs without dependencies
- Parametric *New*
 - Set of similar jobs whose jdl attributes are parametrized
- DAG *New*
 - Group of dependent jobs

Bulk Job Submission

- Bulk job submission allows users to submit multiple jobs (i.e. collections, parametrized jobs and DAG jobs) in one go with obvious advantages:
 - Reduced server connection overhead: the submission is made only once
 - Sandbox sharing
 - Single handle to follow all the jobs in addition to a handle per job to follow them singularly

Bulk Match Making and Shared sandboxes

- Jobs within a **Bulk Job Submission** are likely to share the same requirements. Therefore the Match Making is also done only once. Nodes of the same type are grouped together for this to happen there are two conditions
 - The server must have the feature enabled in the configuration file
 - The user has to set **SignificantAttributes** parameter in the JDL
 - This allows to group nodes by the listed attributes
- They are also likely to share file in the Input SandBox
 - The JDL libs can now identify if there are common files and rearrange the requirement to minimize the amount of data transfer
 - Compression of SandBoxes is also supported

File's perusal

- Allows to inspect job's file while the job is still running it is obtained running together with the job a process on the remote resources.
 - It can be enabled/disabled at anytime after job submission
 - The files to look at can be changed with a WMPProxy command
 - It shouldn't be overused
 - It's certainly useful for debugging

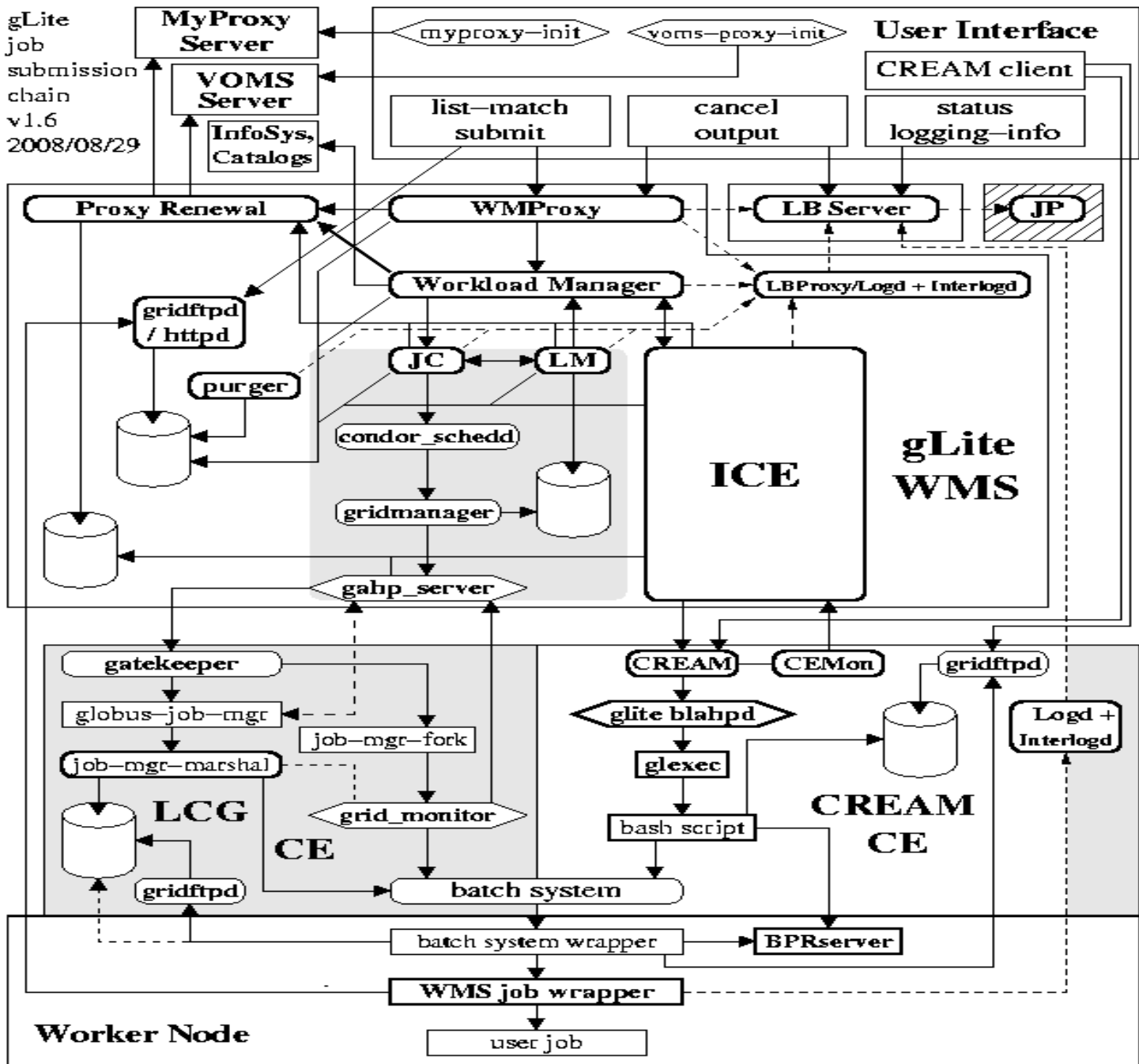
Failover mechanism

- Another interesting feature as many glite clients do fail because the services they contact are temporarily unavailable.
 - A list of WMPproxies needs to be inserted in the client side.
 - The client cannot find the first instance and tries again with the second one and so on
- This works until the job is registered
 - Once the job is registered if the service goes down there's nothing that can be done.
 - So failover is only for the following commands
 - glite-wms-delegate-proxy
 - glite-wms-job-list-match
 - glite-wms-job-submit

Documentation Links

- EGEE documentation page
 - <http://glite.web.cern.ch/glite/documentation/default.asp>
- WMS User Guide
 - <https://edms.cern.ch/file/572489/1/WMS-guide.pdf>
- WMPProxy guide
 - <https://edms.cern.ch/file/674643/1/WMPROXY-guide.pdf>
- JDL WMPProxy job submission (latest version as we speak)
 - <https://edms.cern.ch/file/590869/1/EGEE-JRA1-TEC-590869>
- Glite3 User Guide
 - <https://edms.cern.ch/file/722398/1.2/gLite-3-UserGuide.pdf>

A scary vision



(M. Litmaath)

Questions